Logistic Regression - Patient Readmissions

Patrick Herlihy

12/20/2022

In [19]:
```
import pandas as pd
from statsmodels.formula.api import logit
from sklearn import linear_model
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import plotnine as p9
import seaborn as sns
from statsmodels.graphics.mosaicplot import mosaic
```

Data Cleaning/Preparation

- Import the data from csv to dataframe.
- Check for null values and duplicates.
- Relabel the improperly labeled columns.
- Drop Columns Irrelevant to the analysis
- Review the summary of statistics.
- Explore the dataset and generate visualizations using plotnine.
- Perform ordinal encoding on categorical columns.
- Export the prepared data to csv.

In [20]:
```
#Reads CSV to data frame, sets case order to index
med_predict= pd.read_csv('/Users/herlihpj/Desktop/Data Analytics/D208 - Predictive Modeling/Task 2/medical_clean.csv',
                         index_col=0)
```

In [21]:
```
#Check for duplicated data
duplicates=med_predict.duplicated()
```

```
print('Duplicates: ;', duplicates.sum())
```

```
Duplicates: ; 0
```

In [22]:
```
#Add column titles to last 8 columns of the data frame
med_predict.rename(columns = {'Item1':'Timely admission',
    'Item2':'Timely treatment',
    'Item3':'Timely visits',
    'Item4':'Reliability',
    'Item5':'Options',
    'Item6':'Hours of treatment',
    'Item7':'Courteous staff',
    'Item8':'Evidence of active listening'},
inplace=True)
```

In [23]:
```
# Dropping columns not relavant to the analysis
med_predict = med_predict.drop(columns= ["Customer_id", "Interaction",'TimeZone', "UID", "City",'State', "County", "Zip",
                            "Lng",'Job', "Population",'Additional_charges','TotalCharge'])
```

In [24]:
```
#Explore the data/statistics
print(med_predict.head())
print(med_predict.describe())
print(med_predict.info())
```

```
                Area  Children  Age    Income   Marital  Gender ReAdmis  \
CaseOrder
1           Suburban         1   53  86575.93  Divorced    Male      No
2              Urban         3   51  46805.99   Married  Female      No
3           Suburban         3   53  14370.14   Widowed  Female      No
4           Suburban         0   78  39741.49   Married    Male      No
5              Rural         1   22   1209.56   Widowed  Female      No

            VitD_levels  Doc_visits  Full_meals_eaten  ...     Services  \
CaseOrder                                              ...
1             19.141466           6                 0  ...   Blood Work
2             18.940352           4                 2  ...  Intravenous
3             18.057507           4                 1  ...   Blood Work
4             16.576858           4                 1  ...   Blood Work
5             17.439069           5                 0  ...      CT Scan

           Initial_days Timely admission Timely treatment Timely visits  \
CaseOrder
1             10.585770               3                3             2
2             15.129562               3                4             3
3              4.772177               2                4             4
4              1.714879               3                5             5
5              1.254807               2                1             3

           Reliability Options Hours of treatment Courteous staff  \
CaseOrder
1                    2       4                 3               3
2                    4       4                 4               3
3                    4       3                 4               3
4                    3       4                 5               5
5                    3       5                 3               4

           Evidence of active listening
CaseOrder
1                                     4
2                                     3
3                                     3
4                                     5
5                                     3

[5 rows x 35 columns]
            Children           Age         Income   VitD_levels    Doc_visits  \
count  10000.000000  10000.000000   10000.000000  10000.000000  10000.000000
mean       2.097200     53.511700   40490.495160     17.964262      5.012200
std        2.163659     20.638538   28521.153293      2.017231      1.045734
```

```
min        0.000000     18.000000     154.080000      9.806483      1.000000
25%        0.000000     36.000000   19598.775000     16.626439      4.000000
50%        1.000000     53.000000   33768.420000     17.951122      5.000000
75%        3.000000     71.000000   54296.402500     19.347963      6.000000
max       10.000000     89.000000  207249.100000     26.394449      9.000000

       Full_meals_eaten     vitD_supp  Initial_days  Timely admission  \
count      10000.000000  10000.000000  10000.000000      10000.000000
mean           1.001400      0.398900     34.455299          3.518800
std            1.008117      0.628505     26.309341          1.031966
min            0.000000      0.000000      1.001981          1.000000
25%            0.000000      0.000000      7.896215          3.000000
50%            1.000000      0.000000     35.836244          4.000000
75%            2.000000      1.000000     61.161020          4.000000
max            7.000000      5.000000     71.981490          8.000000

       Timely treatment  Timely visits   Reliability       Options  \
count      10000.000000   10000.000000  10000.000000  10000.000000
mean           3.506700       3.511100      3.515100      3.496900
std            1.034825       1.032755      1.036282      1.030192
min            1.000000       1.000000      1.000000      1.000000
25%            3.000000       3.000000      3.000000      3.000000
50%            3.000000       4.000000      4.000000      3.000000
75%            4.000000       4.000000      4.000000      4.000000
max            7.000000       8.000000      7.000000      7.000000

       Hours of treatment  Courteous staff  Evidence of active listening
count        10000.000000     10000.000000                  10000.000000
mean             3.522500         3.494000                      3.509700
std              1.032376         1.021405                      1.042312
min              1.000000         1.000000                      1.000000
25%              3.000000         3.000000                      3.000000
50%              4.000000         3.000000                      3.000000
75%              4.000000         4.000000                      4.000000
max              7.000000         7.000000                      7.000000
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 10000
Data columns (total 35 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Area                         10000 non-null  object
 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
 27  Timely admission              10000 non-null  int64
 28  Timely treatment              10000 non-null  int64
 29  Timely visits                 10000 non-null  int64
 30  Reliability                   10000 non-null  int64
 31  Options                       10000 non-null  int64
 32  Hours of treatment            10000 non-null  int64
 33  Courteous staff               10000 non-null  int64
 34  Evidence of active listening  10000 non-null  int64
dtypes: float64(3), int64(13), object(19)
memory usage: 2.7+ MB
None
```

In [25]:

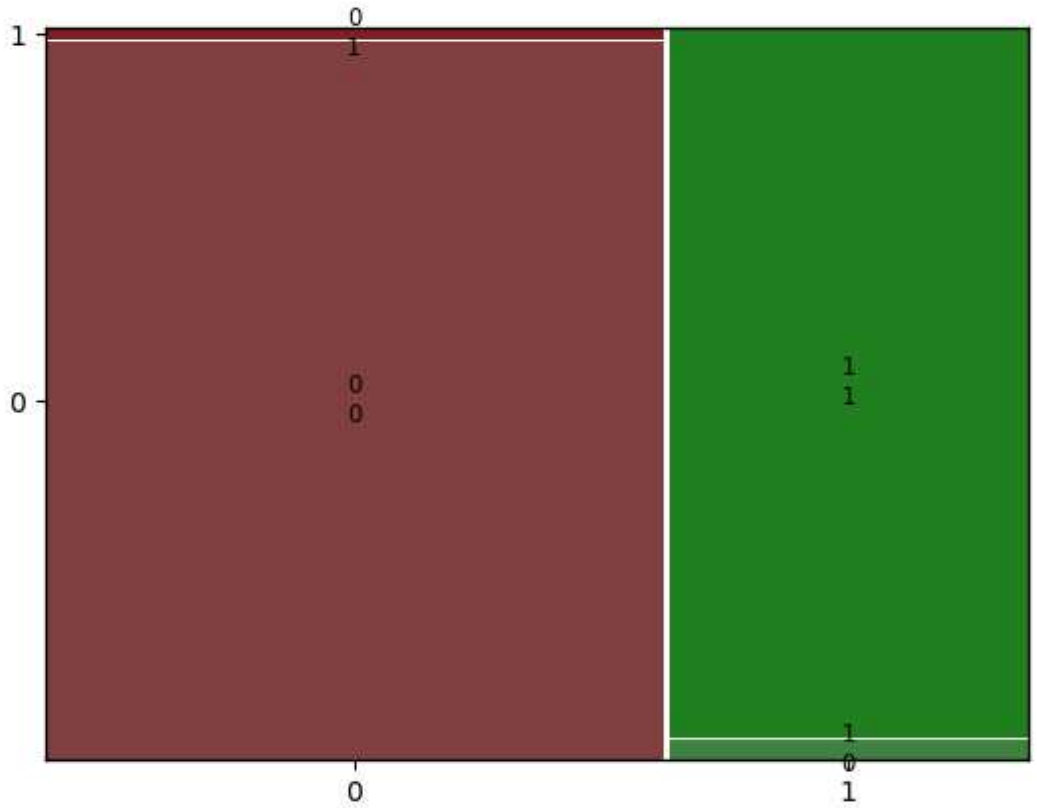
```
#Generate mean values by category to view relationship
print('Mean values of Variables by ReAdmission: ')
#Initial Days
idayssummarystats=med_predict.groupby("ReAdmis")["Initial_days"].mean()
print(idayssummarystats)
#Age
agesummarystats=med_predict.groupby("ReAdmis")["Age"].mean()
print(agesummarystats)
#Children
childrensummarystats=med_predict.groupby("ReAdmis")["Children"].mean()
print(childrensummarystats)
```

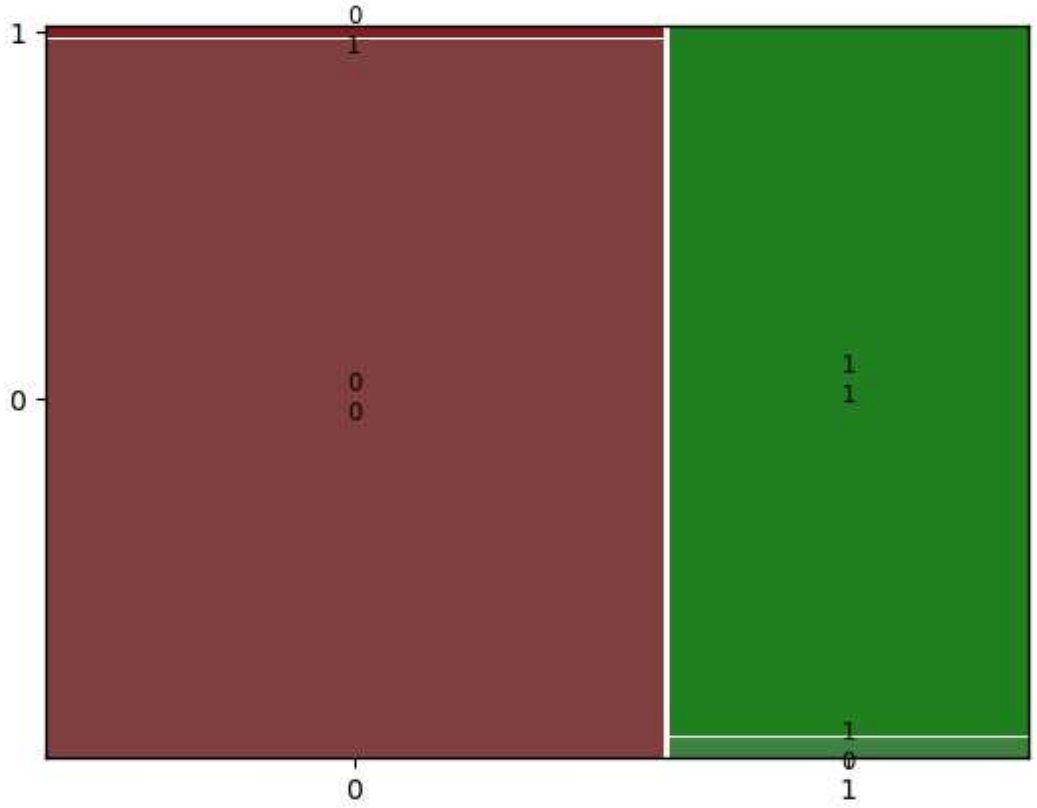
```
#Stroke
strokesummarystats=med_predict.groupby("ReAdmis")["Stroke"].value_counts()
print(strokesummarystats)
```

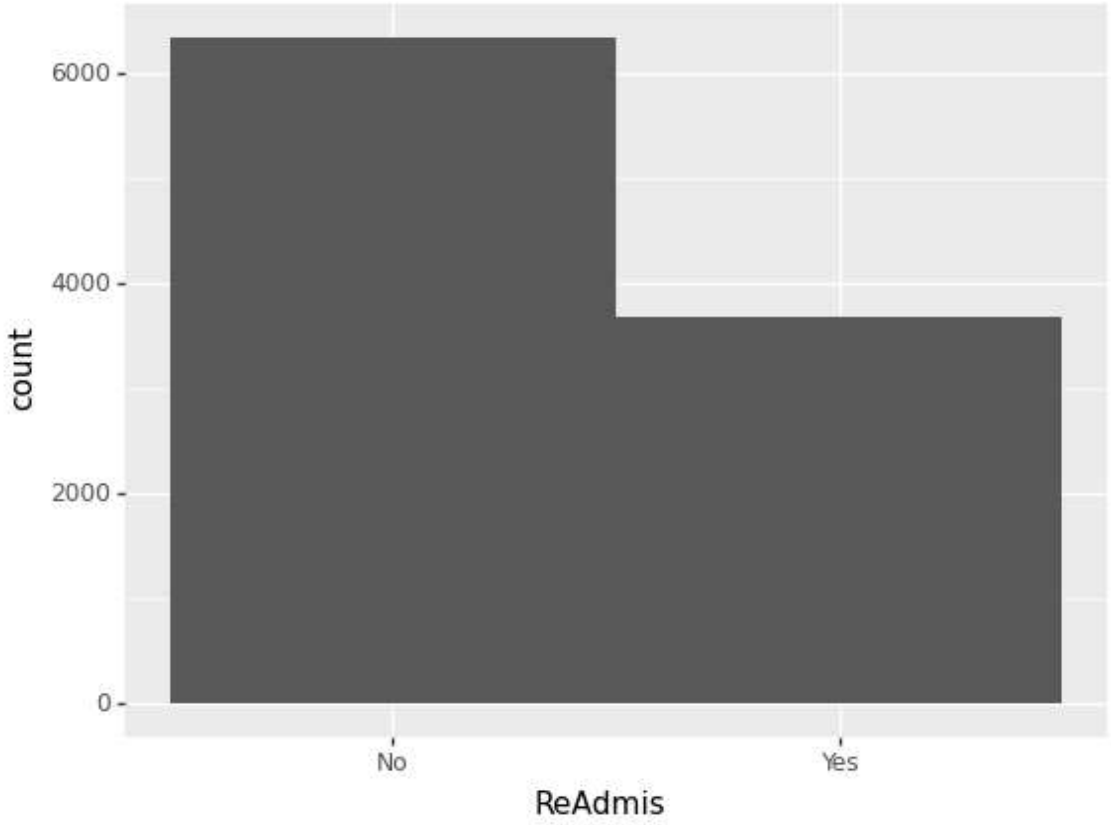
```
Mean values of Variables by ReAdmission:
ReAdmis
No     17.414699
Yes    63.859507
Name: Initial_days, dtype: float64
ReAdmis
No     53.263308
Yes    53.940311
Name: Age, dtype: float64
ReAdmis
No     2.058443
Yes    2.164077
Name: Children, dtype: float64
ReAdmis  Stroke
No       No        5071
         Yes       1260
Yes      No        2936
         Yes        733
Name: Stroke, dtype: int64
```

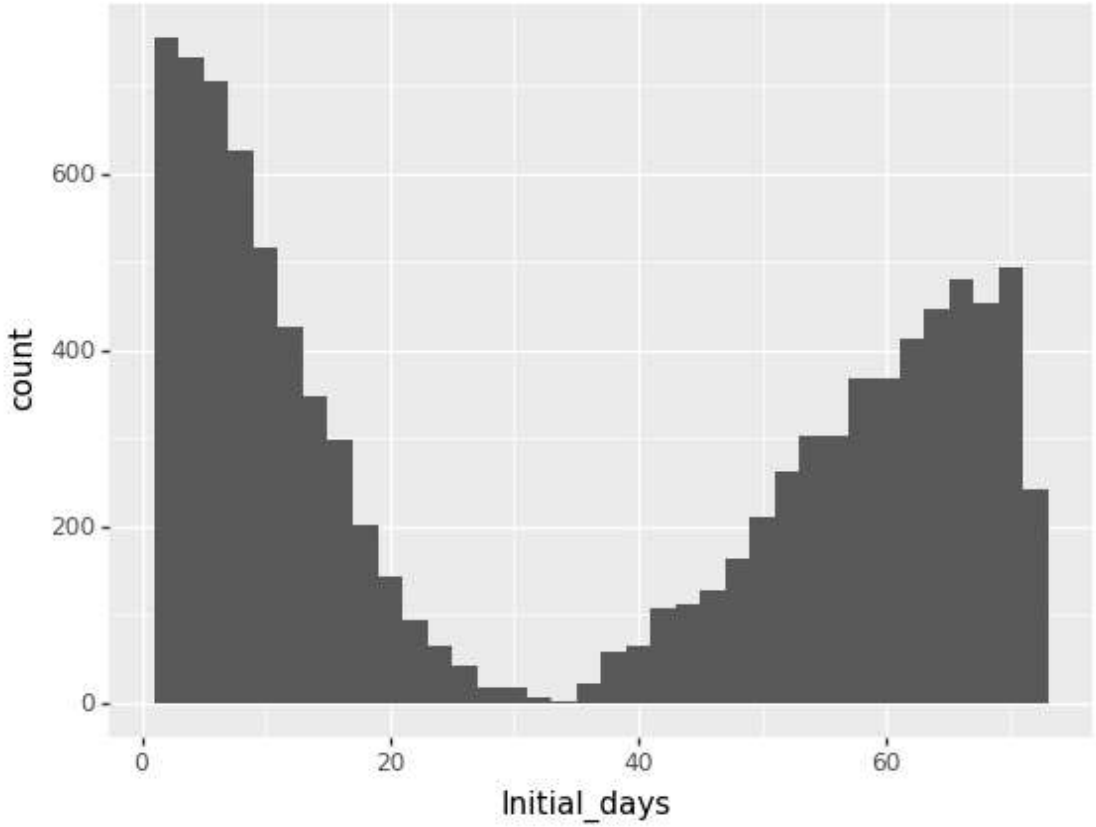
In [26]:

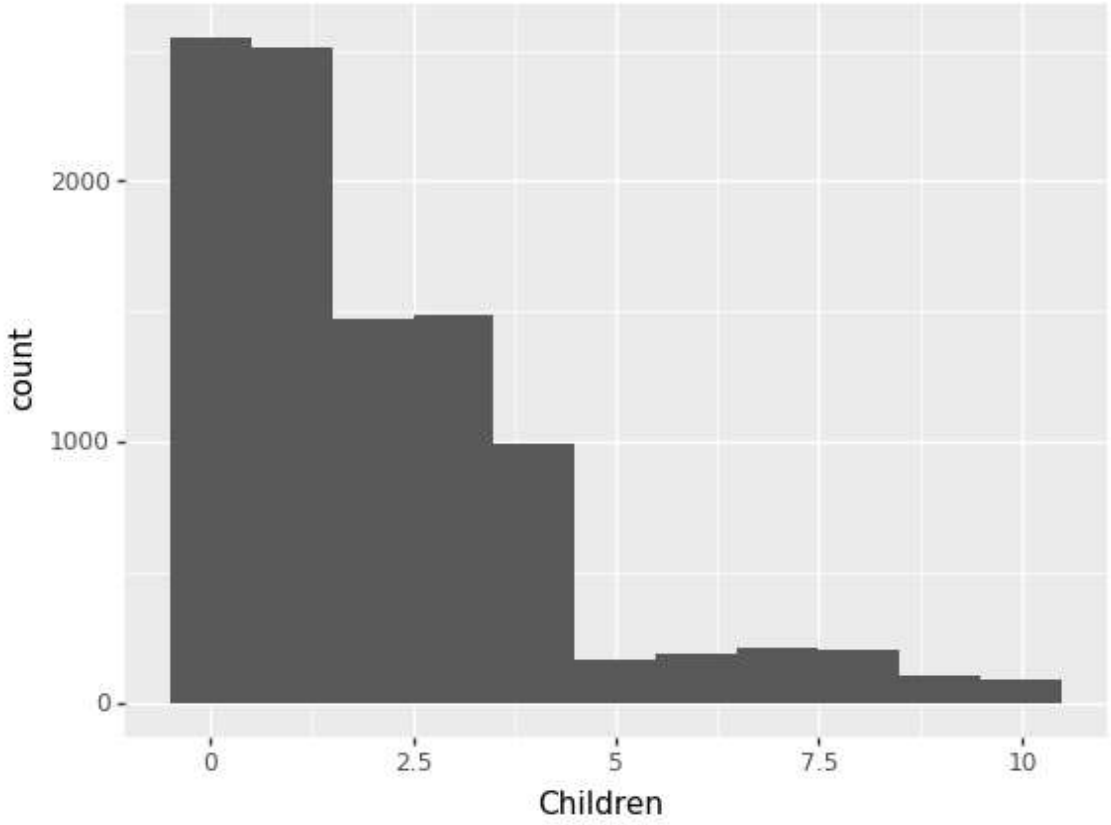
```
#Univariate
#Readmis
print(p9.ggplot(med_predict)+p9.aes(x='ReAdmis')+ p9.geom_histogram(binwidth=1))
#Initial_days
print(p9.ggplot(med_predict)+p9.aes(x='Initial_days')+ p9.geom_histogram(binwidth=2))
#children
print(p9.ggplot(med_predict)+p9.aes(x='Children')+ p9.geom_histogram(binwidth=1))
#Age
print(p9.ggplot(med_predict)+p9.aes(x='Age')+ p9.geom_histogram(binwidth=5))
#Categorical Predictors
symptoms=['Asthma', 'Stroke', 'Arthritis', 'Anxiety',]
for s in symptoms:
    print(p9.ggplot(med_predict)+p9.aes(x=s)+ p9.geom_histogram(binwidth=1))

#Bivariate
#Categorical
servicebyreadmission=(p9.ggplot(med_predict)+ p9.aes(x='Services',y='Initial_days', fill='ReAdmis')+ p9.geom_boxplot())
strokebyreadmission=(p9.ggplot(med_predict)+ p9.aes(x='Stroke',y='Initial_days', fill='ReAdmis')+ p9.geom_boxplot())
#Continuous
daysvsageonstroke=p9.ggplot(med_predict)+ p9.aes(y='Initial_days',x='Age', color='Stroke')+ p9.geom_point()
daysvsage=p9.ggplot(med_predict)+ p9.aes(y='Initial_days',x='Age', color='ReAdmis')+ p9.geom_point()
print(servicebyreadmission,strokebyreadmission,daysvsageonstroke,daysvsage)
```

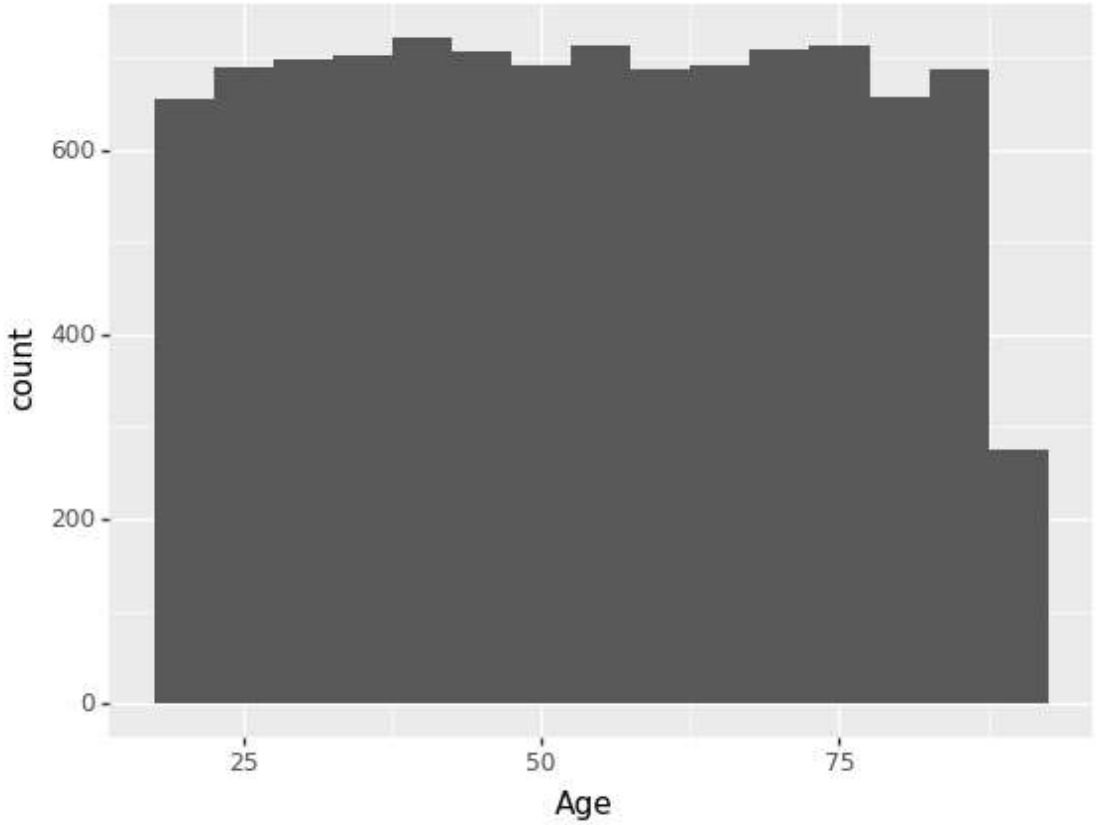

1
0
0
1
0
0
1
1
1
0
0
1

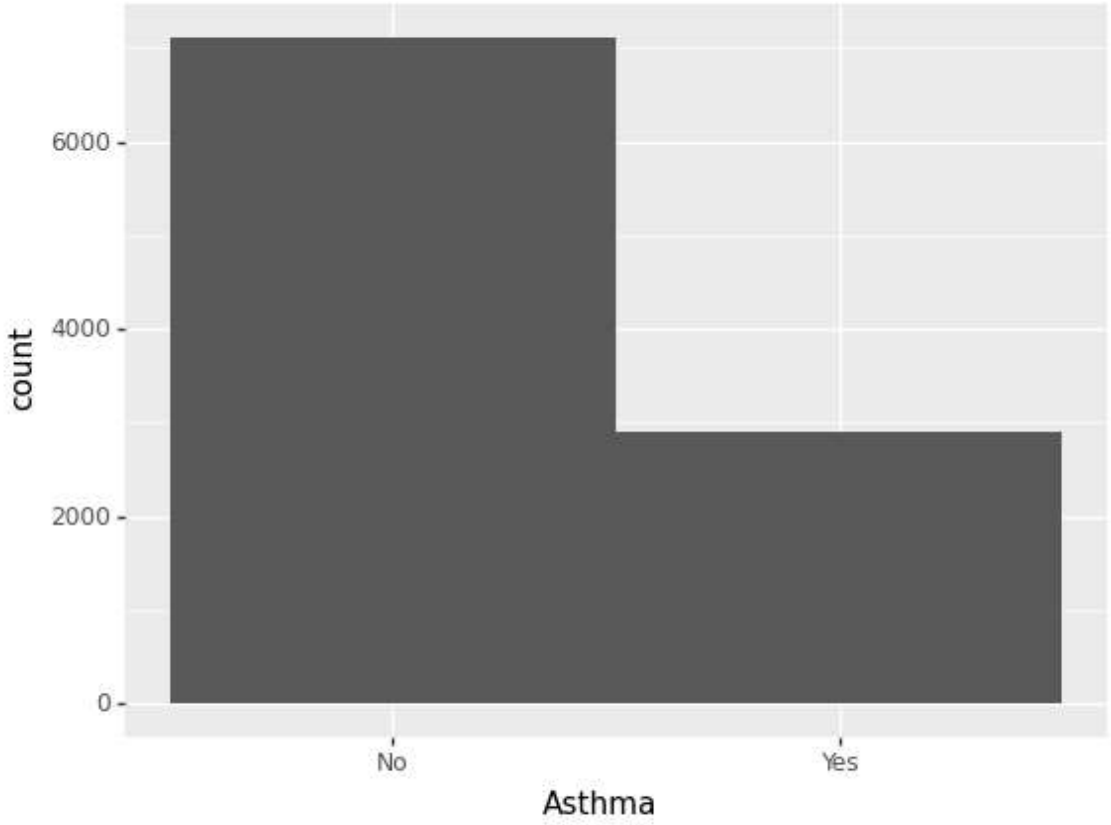

1
0
0
1
0
0
1
1
1
1
0
0
1

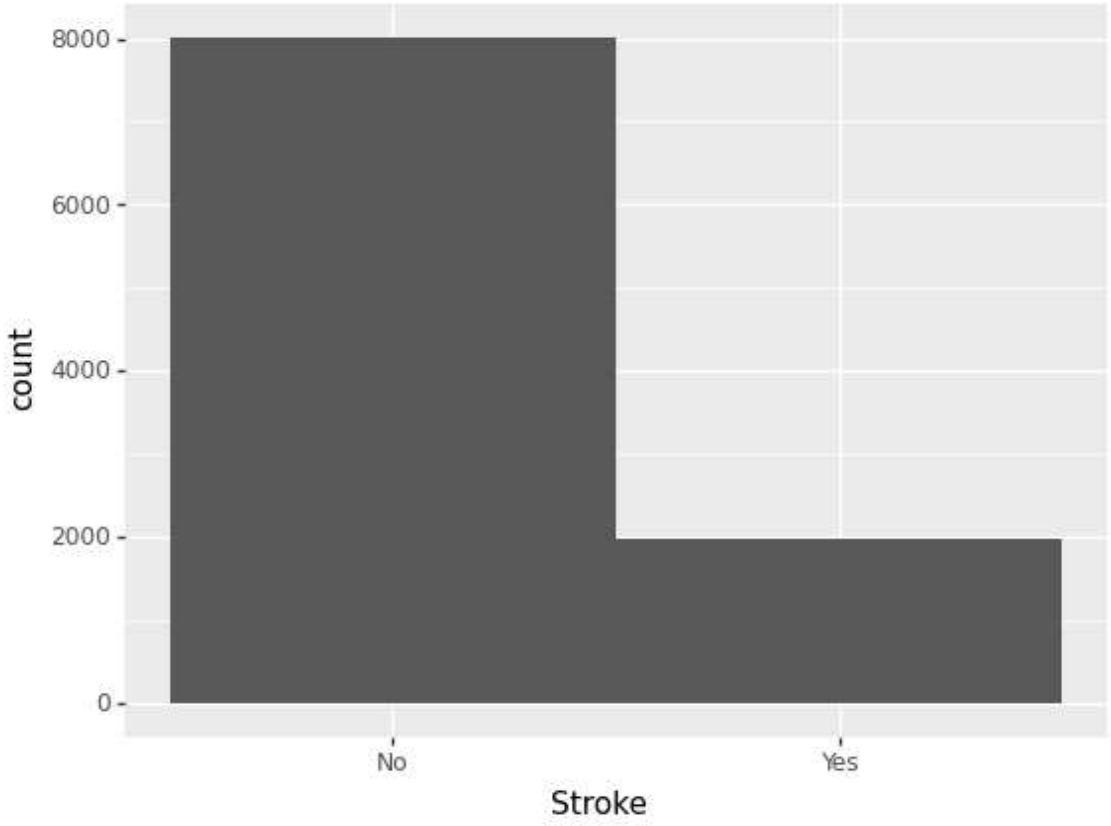

6000
4000
2000
0
count
No
Yes
ReAdmis

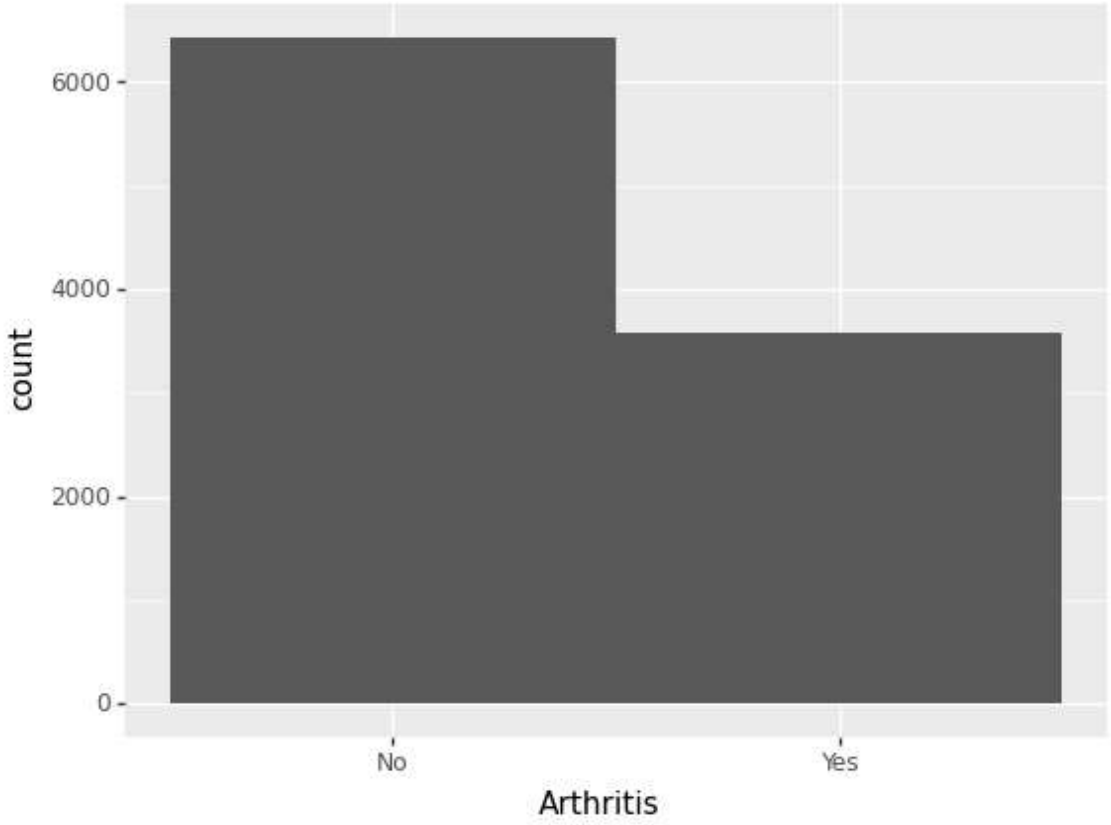

count
600
400
200
0
0
20
40
60
Initial_days

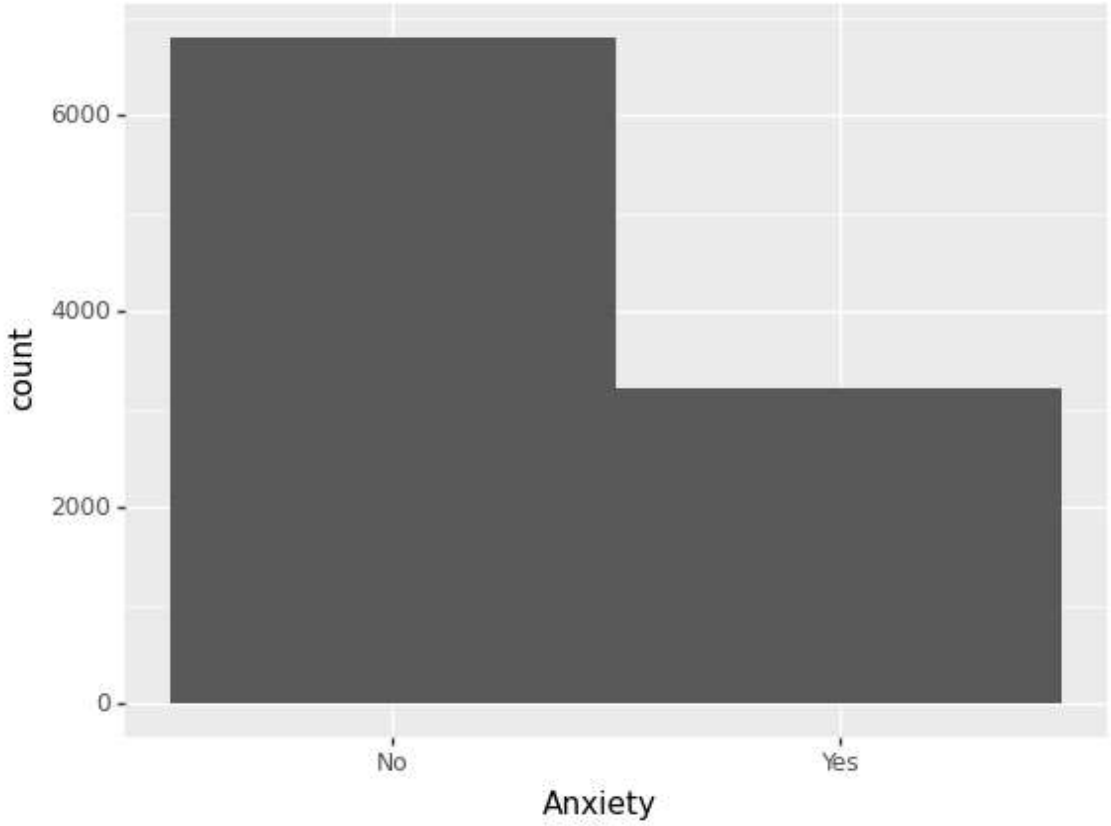

2000
1000
0
count
0
2.5
5
7.5
10
Children

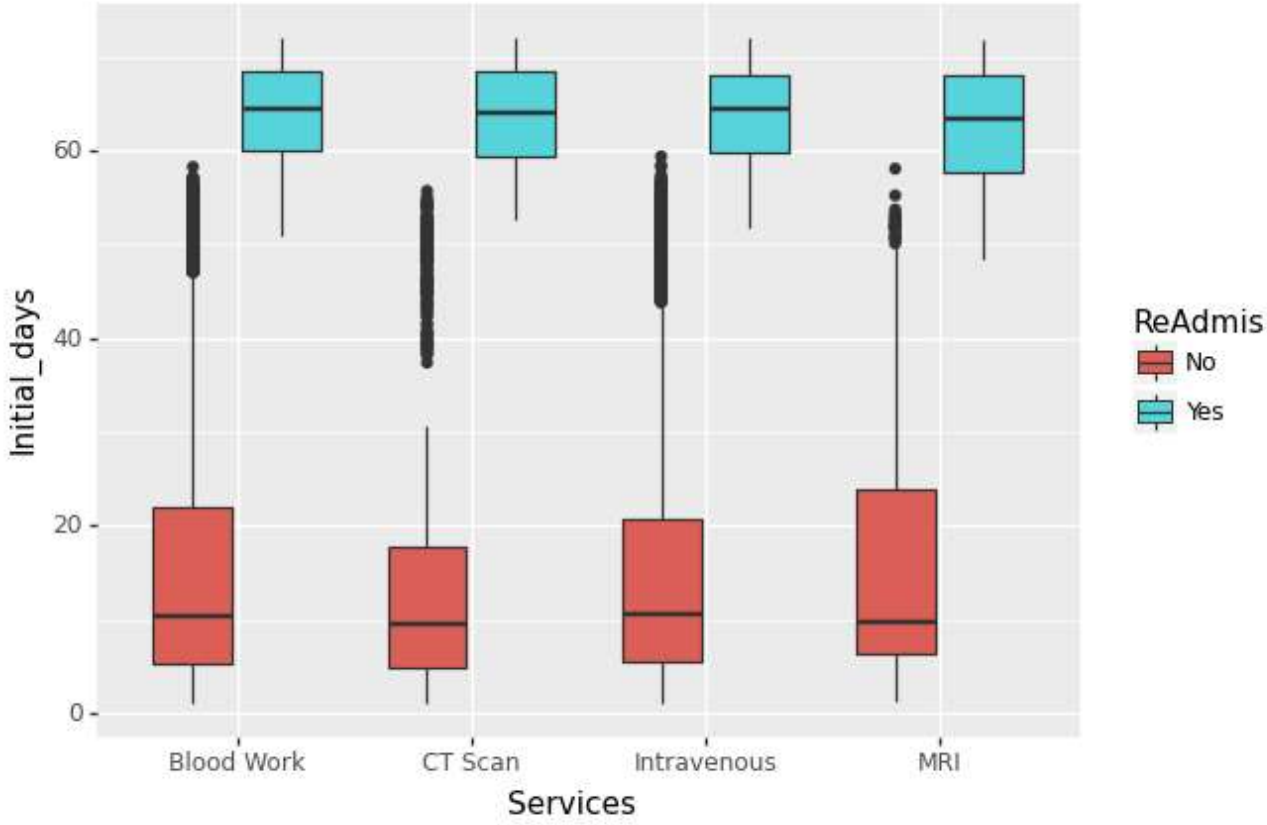

600
400
200
0
count
25
50
75
Age

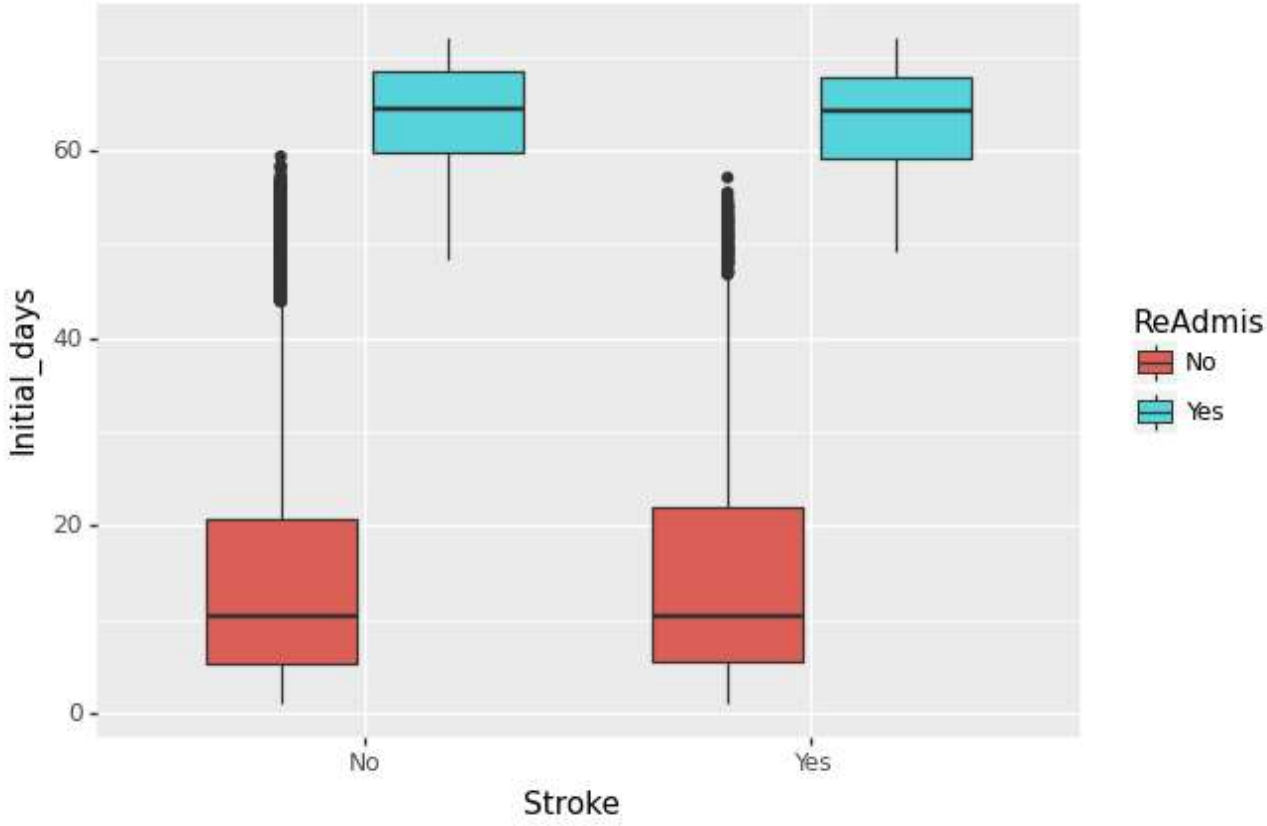

6000
4000
2000
0
count
No
Yes
Asthma

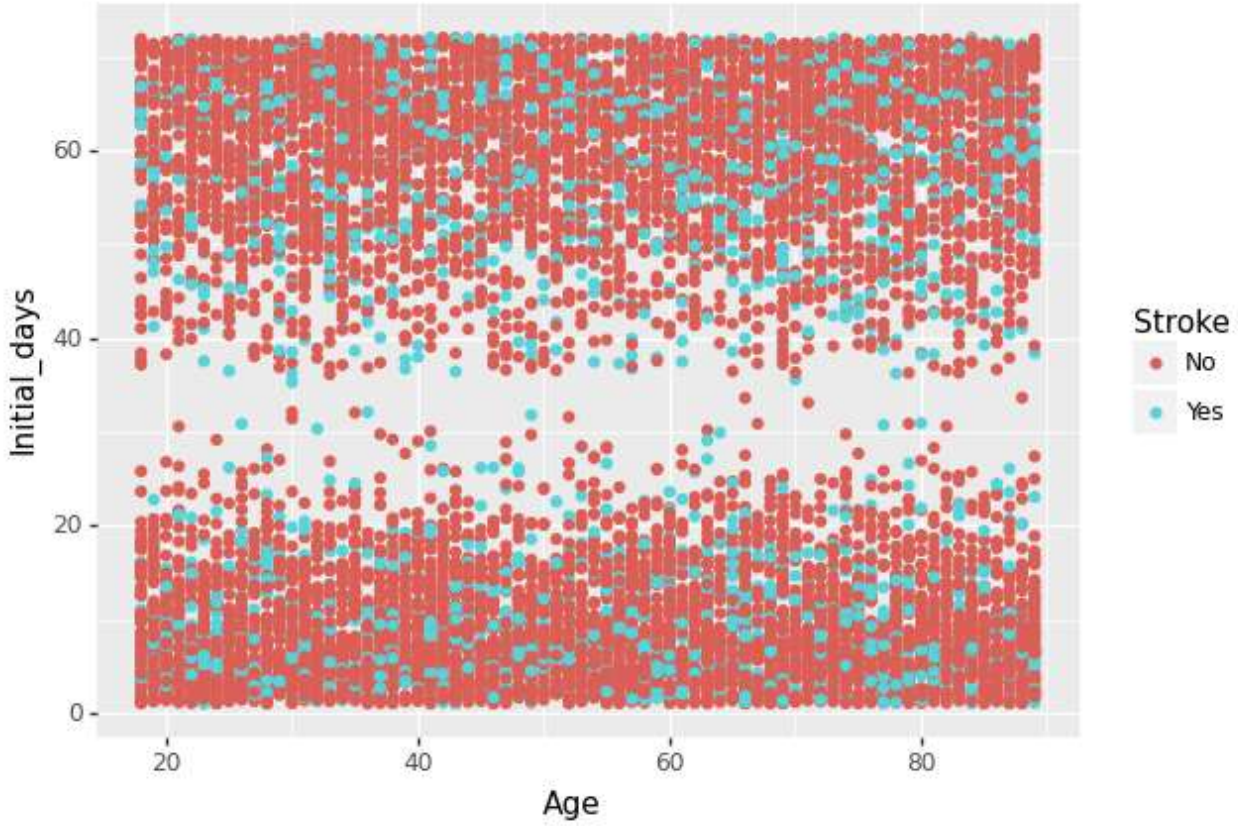

8000
6000
4000
2000
0
count
No
Yes
Stroke

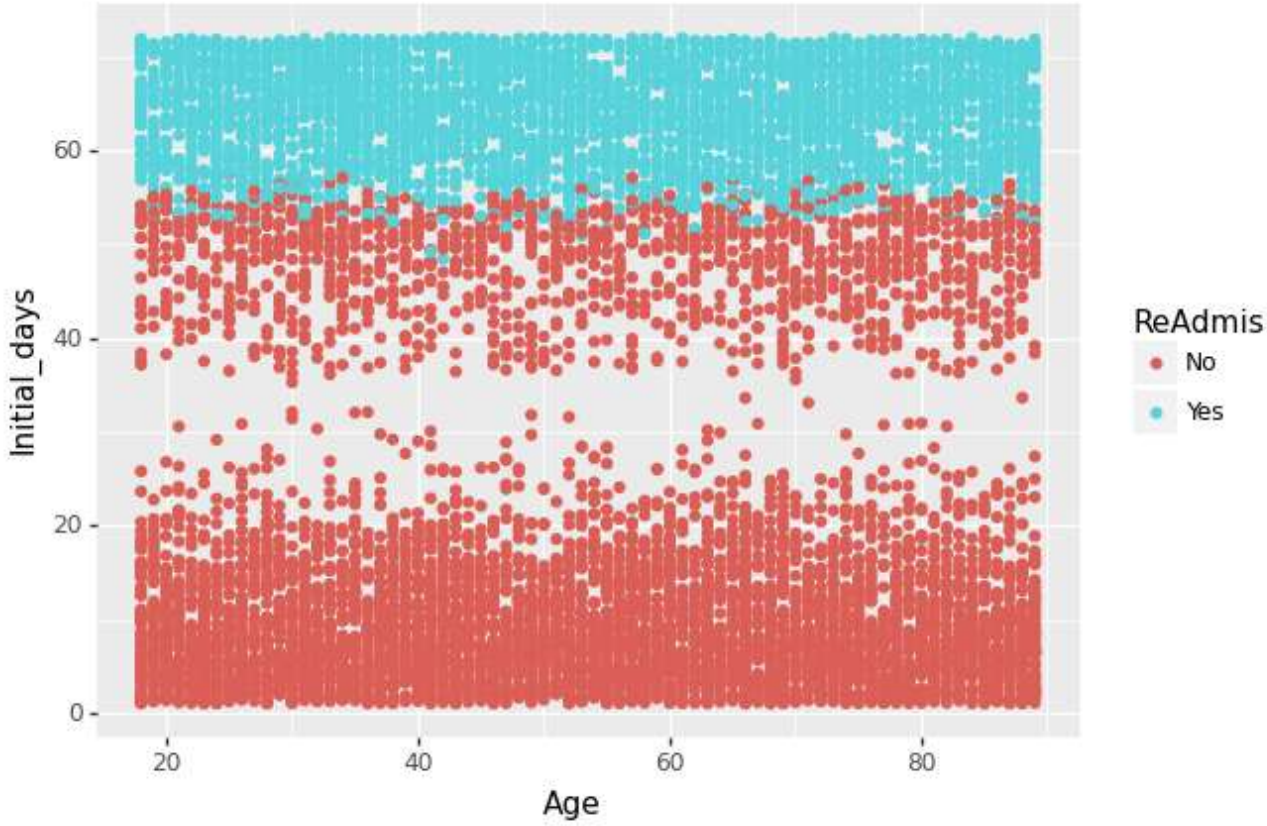

6000
4000
2000
0
count
No
Yes
Arthritis


6000
4000
2000
0
count
No
Yes
Anxiety


Initial_days
60
40
20
0
Blood Work
CT Scan
Intravenous
MRI
Services
ReAdmis
No
Yes


Initial_days
60
40
20
0
No
Yes
Stroke
ReAdmis
No
Yes


Initial_days
60
40
20
0
20
40
60
80
Age
Stroke
No
Yes



In [27]:
```
#Ordinal Encoding to convert to numeric 0:No, 1:Yes; other variable alphabetically starting with 0
oe_dict={}
#list of columns to convert to numerical
convert_cols=['Area','Services', 'Marital', 'Gender','ReAdmis','Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
              'Complication_risk','Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia','BackPain', 'Anxiety',
             'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services']
for col_name in convert_cols:
    print(col_name+' pre: '+str(med_predict[col_name].unique()))
    #Creates column ordinal encoder
    oe_dict[col_name]=OrdinalEncoder()
    col=med_predict[col_name]
    #select non-null values of col
    col_not_null=col[col.notnull()]
    reshaped_vals=col_not_null.values.reshape(-1,1)
    encoded_vals=oe_dict[col_name].fit_transform(reshaped_vals)
    med_predict.loc[col.notnull(), col_name]=np.squeeze(encoded_vals)
    print(col_name+' post: '+str(med_predict[col_name].unique()))
```

```
Area pre: ['Suburban' 'Urban' 'Rural']
Area post: [1. 2. 0.]
Services pre: ['Blood Work' 'Intravenous' 'CT Scan' 'MRI']
Services post: [0. 2. 1. 3.]
Marital pre: ['Divorced' 'Married' 'Widowed' 'Never Married' 'Separated']
Marital post: [0. 1. 4. 2. 3.]
Gender pre: ['Male' 'Female' 'Nonbinary']
Gender post: [1. 0. 2.]
ReAdmis pre: ['No' 'Yes']
ReAdmis post: [0. 1.]
Soft_drink pre: ['No' 'Yes']
Soft_drink post: [0. 1.]
Initial_admin pre: ['Emergency Admission' 'Elective Admission' 'Observation Admission']
Initial_admin post: [1. 0. 2.]
HighBlood pre: ['Yes' 'No']
HighBlood post: [1. 0.]
Stroke pre: ['No' 'Yes']
Stroke post: [0. 1.]
Complication_risk pre: ['Medium' 'High' 'Low']
Complication_risk post: [2. 0. 1.]
Overweight pre: ['No' 'Yes']
Overweight post: [0. 1.]
Arthritis pre: ['Yes' 'No']
Arthritis post: [1. 0.]
Diabetes pre: ['Yes' 'No']
Diabetes post: [1. 0.]
Hyperlipidemia pre: ['No' 'Yes']
Hyperlipidemia post: [0. 1.]
BackPain pre: ['Yes' 'No']
BackPain post: [1. 0.]
Anxiety pre: ['Yes' 'No']
Anxiety post: [1. 0.]
Allergic_rhinitis pre: ['Yes' 'No']
Allergic_rhinitis post: [1. 0.]
Reflux_esophagitis pre: ['No' 'Yes']
Reflux_esophagitis post: [0. 1.]
Asthma pre: ['Yes' 'No']
Asthma post: [1. 0.]
Services pre: [0. 2. 1. 3.]
Services post: [0. 2. 1. 3.]
```

In [11]:
```
med_predict.to_csv('/Users/herlihpj/Desktop/Data Analytics/D208 - Predictive Modeling/Task 2/Prepared_data.csv')
```

Begin Analysis

- Define functions
- Perform Multiple Logistic Regression for patient readmission on all variables
- Utilize forward stepwise selection to determine the top 3 variables on patient readmission
- Construct a final model using those 3 variables
- Review model results using a confusion matrix

In [28]:
```
##### Functions ######
#Builds multiple logistic regression formula
def formula_builder(target_variable, predictor_variables):
    formula=target_variable+' ~ '
    end=len(predictor_variables)-1
    #use enumerate to get first and last for string manipulation
    for count, value in enumerate(predictor_variables):
        if count==end:
            formula=formula+value
        else:
            formula=formula+value+' + '
    return formula

#Picking the best variables for use
def auc(variables, target, table):
    X = table[variables]
    y = table[target]
    y = y.values.ravel()
    logreg = linear_model.LogisticRegression()
    logreg.fit(X, y)
    predictions = logreg.predict_proba(X)[:,1]
    auc = roc_auc_score(y, predictions)
    return(auc)

#how next_best works
def next_best(current_variables,candidate_variables, target, table):
    best_auc = -1
    best_variable = None
    for v in candidate_variables:
        auc_v = auc(current_variables + [v], target, table)
        if auc_v >= best_auc:
            best_auc = auc_v
            best_variable = v
    return best_variable
```

In [13]:
```
target = ["ReAdmis"]
target_str=target[0]
#use all columns in the data frame to find the best candidates
#candidate_variables = list(med_predict.columns.values)
#or use preset list
candidate_variables=['Asthma','Area','Services','Marital','Gender','Soft_drink', 'Initial_admin', 'ReAdmis', 'Children',
                     'HighBlood', 'Stroke', 'Complication_risk','Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia',
                     'Anxiety','Allergic_rhinitis', 'Reflux_esophagitis','Age', 'VitD_levels','Doc_visits','vitD_supp',
if target_str in candidate_variables:
    candidate_variables.remove(target_str)
    print('Target Variable was removed')
#Initial Model
ireadmis_model=logit(formula_builder(target_str, candidate_variables), data=med_predict).fit()
print(ireadmis_model.summary())
```

```
Target Variable was removed
Optimization terminated successfully.
         Current function value: 0.041778
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                10000
Model:                           Logit   Df Residuals:                     9975
Method:                            MLE   Df Model:                           24
Date:                Mon, 27 Mar 2023   Pseudo R-squ.:                  0.9364
Time:                        22:17:09   Log-Likelihood:                -417.78
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept            -65.4086      3.395    -19.265      0.000     -72.063     -58.754
Asthma                -0.9872      0.201     -4.906      0.000      -1.382      -0.593
Area                1.705e-05      0.112      0.000      1.000      -0.220       0.220
Services              0.1659      0.089      1.864      0.062      -0.009       0.340
Marital               0.0370      0.064      0.574      0.566      -0.089       0.163
Gender                0.0728      0.166      0.438      0.661      -0.253       0.398
Soft_drink            0.0646      0.208      0.310      0.757      -0.344       0.473
Initial_admin         0.3065      0.126      2.429      0.015       0.059       0.554
Children              0.0699      0.041      1.721      0.085      -0.010       0.149
HighBlood             0.7786      0.190      4.102      0.000       0.407       1.151
Stroke                1.4444      0.237      6.095      0.000       0.980       1.909
Complication_risk    -0.1160      0.102     -1.135      0.257      -0.316       0.084
Overweight           -0.1690      0.199     -0.849      0.396      -0.559       0.221
Arthritis            -0.9220      0.194     -4.751      0.000      -1.302      -0.542
Diabetes              0.4318      0.201      2.149      0.032       0.038       0.826
Hyperlipidemia        0.3704      0.192      1.934      0.053      -0.005       0.746
BackPain              0.3111      0.182      1.710      0.087      -0.045       0.668
Anxiety              -0.7885      0.195     -4.041      0.000      -1.171      -0.406
Allergic_rhinitis    -0.2315      0.184     -1.255      0.210      -0.593       0.130
Reflux_esophagitis   -0.2332      0.185     -1.260      0.208      -0.596       0.129
Age                  -0.0009      0.004     -0.211      0.833      -0.009       0.008
VitD_levels           0.0430      0.044      0.985      0.325      -0.043       0.129
Doc_visits            0.0198      0.084      0.237      0.813      -0.144       0.184
vitD_supp            -0.0639      0.139     -0.459      0.646      -0.337       0.209
Initial_days          1.1823      0.059     19.912      0.000       1.066       1.299
==============================================================================

Possibly complete quasi-separation: A fraction 0.78 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 [29]:
```
### Forward Stepwise variable selection ###
current_variables = []

max_number_variables = 3
number_iterations = min(max_number_variables, len(candidate_variables))

for i in range(0,number_iterations):
    next_variable = next_best(current_variables,candidate_variables,target,med_predict)
    current_variables = current_variables + [next_variable]
    candidate_variables.remove(next_variable)
    #show AUC Score by step
    model_auc=auc(current_variables,target, med_predict)
    print('Step ', i+1,' AUC Score: ',round(model_auc,3))
print('Top ',max_number_variables,' variables for a '+ target_str+ ' Logistic Regression model are: ')
print(current_variables) #returns the top variables
```

```
Step  1  AUC Score:  0.512
Step  2  AUC Score:  0.519
Step  3  AUC Score:  0.521
Top  3  variables for a ReAdmis Logistic Regression model are:
['Children', 'Age', 'vitD_supp']
```

In [15]:
```
#Final Reduced Model
freadmis_model=logit(formula_builder(target_str, current_variables), data=med_predict).fit()
print(freadmis_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.045875
         Iterations 13
                           Logit Regression Results
==============================================================================
Dep. Variable:                 ReAdmis   No. Observations:                10000
Model:                           Logit   Df Residuals:                     9996
Method:                            MLE   Df Model:                            3
Date:                 Mon, 27 Mar 2023   Pseudo R-squ.:                  0.9302
Time:                         22:19:26   Log-Likelihood:                -458.75
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -57.6984      2.734    -21.108      0.000     -63.056     -52.341
Initial_days   1.0628      0.050     21.148      0.000       0.964       1.161
Stroke         1.2572      0.223      5.650      0.000       0.821       1.693
Asthma        -0.9532      0.189     -5.038      0.000      -1.324      -0.582
==============================================================================

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.
```

In [31]:
```
#Check final model agains a sklearn model
print('logit')
final_model=linear_model.LogisticRegression()
X=med_predict[current_variables]
Y=med_predict[target]
Y = Y.values.ravel()
final_model.fit(X,Y)
print(final_model.coef_)
print(final_model.intercept_)
```

```
logit
[[0.02227192 0.00155589 0.03606204]]
[-0.69038783]
```

In [33]:
```
#Confusion Matrix and model Accuracy
conf_matrix = freadmis_model.pred_table()
print(conf_matrix)

# Extract TN, TP, FN and FP from conf_matrix
TN = conf_matrix[0,0]
TP = conf_matrix[1,1]
```

```
FN = conf_matrix[1,0]
FP = conf_matrix[0,1]

# Calculate and print the accuracy
accuracy = (TN + TP) / (TN + FN + FP + TP)
print("accuracy", round(accuracy,3))

# Calculate and print the sensitivity
sensitivity = TP / (TP + FN)
print("sensitivity", round(sensitivity,3))

# Calculate and print the specificity
specificity = TN / (TN + FP)
print("specificity", round(specificity,3))

#Creates the confusion matrix mosaic
print(mosaic(conf_matrix))
```

```
[[6227.  104.]
 [ 103. 3566.]]
accuracy 0.979
sensitivity 0.972
specificity 0.984
(<Figure size 640x480 with 3 Axes>, {('0', '0'): (0.0, 0.0, 0.629950248756219, 0.9803052112397415), ('0', '1'): (0.0, 0.
9836274703759541, 0.629950248756219, 0.01637252962404581), ('1', '0'): (0.6349253731343284, 0.0, 0.36507462686567166, 0.
0279797784979477), ('1', '1'): (0.6349253731343284, 0.031302037634160326, 0.36507462686567166, 0.9686979623658398)})
```

Summary of Results

The initial model containing 24 variables had an R-Squared value of 0.9364 which proved there was a correlation between the chosen variables and customer Readmission. After reviewing the summary of the model's P-values it was easy to identify that a large amount of the variance could be explained by the several of the variables. A reduced model containing the top 3 of these variables had an R-squared value of .9302. These variables were chosen using sklearn's roc_auc_score and forward stepwise selection. This shows that when dropping 21 of the variables in the final model, little explanation of the variance was lost.

The final model was found to have an accuracy of ~98% sensitivity of 97% and specitivity of 98%. These values prove that this is an effective model.