Matthew Morgan
Student ID: 010471280
Data Mining - 209
Task 1: Classification Analysis
Western Governor's University

Part I: Research Question
A1. Research Question
Can I predict which patients are at risk of re-admission using the k-nearest neighbor’s so the hospital can take appropriate steps to reduce re-admissions?

A2. Goal of the Data Analysis
The primary goal of this data analysis is to develop a machine learning model using k-NN to help the company identify patients who are at risk of re-admission.

Part II: Method Justification
B1. Explain Method from Part A1

k-NN analysis uses the proximity of other data points to make predictions about unlabeled data points. k-NN commonly uses standard Euclidean distance to determine how far away the nearest labeled neighbors are to the unlabeled neighbor to make a prediction. k-NN was chosen for this task because it can help differentiate between multiple data points by looking for similarities between labeled and unlabeled data points. This will allow us to produce a model to successfully predict patients that have a higher risk for future re-admission.

B2. Summarize one assumption of your chosen classification model.

k-NN assumes that similar things exist near each other, while if a data point is far away from another group, it's dissimilar to those data points. The algorithm depends on this assumption being true enough for the algorithm to be useful. The algorithm classifies new data points based on how the neighbors are classified.

B3. List Packages or Libraries Chosen

Packages	Usage
Pandas	Importing data and data manipulation
Numpy	Provides array objects for calculations
Seaborn	For visualizations like correlation matrix
Matplotlib.pyplot	For visualizations like ROC curve
missingno	For visualizing missing data
Sklearn,preprocessing	To scale features
Skleanr.feature_selection	For feature selection
Sklearn.model_selection	For splitting data into train and test sets
Sklearn.pipeline	To assemble several steps that can be performed together while setting different parameters
Scipy.stats	To run statistical calculations
Statsmodels.formula.api import ols	To calculate linear regression

In [22]:
```
# Data Analytics imports
import pandas as pd
import numpy as np

# Visualization imports
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno

# Statistics imports
import scipy.stats as stats
from scipy.stats import skew, kurtosis
import statistics as stat

# Linear regression import
from statsmodels.formula.api import ols

# scikit-learn imports
import sklearn
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

#Create KNN model
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

#Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

```
Part III: Data Preparation
C1. One Data Preprocessing Goal from A1
Removing white spaces, imputing missing data, converting binary (yes/no) variables into quantitative (1/0) variables.

C2. Identify Initial Data Set Variables Used for Analysis
Variable #  Independent Variable               Data Type   Data Class
         1  Initial_days                       Continuous  Quantitative
         2  Services_CT_Scan                   Categorical Qualitative
         3  Children                           Continuous  Quantitative
         4  Services_Intravenous               Categorical Qualitative
         5  Population                         Continuous  Quantitative
         6  Initial_Admin_Emergency_admission  Categorical Qualitative


C3. Explain Each of the Steps Used to Prepare the Data for Analysis
1.  Load data into dataframe
2.  View the data to evaluate structure and types
3.  Detect null values
4.  Check for missing data
5.  Visualize data to check for outliers
6.  Convert categorical data to quantitative
7.  Rename columns from pd.get_dummies
8.  Visualize univariate stats from dataframe to ensure data quality
```

In [23]:
```
#Loading the CSV of the default dataset
df = pd.read_csv(r'C:\Users\mmorg\WGU\D209\medical_clean.csv')
```

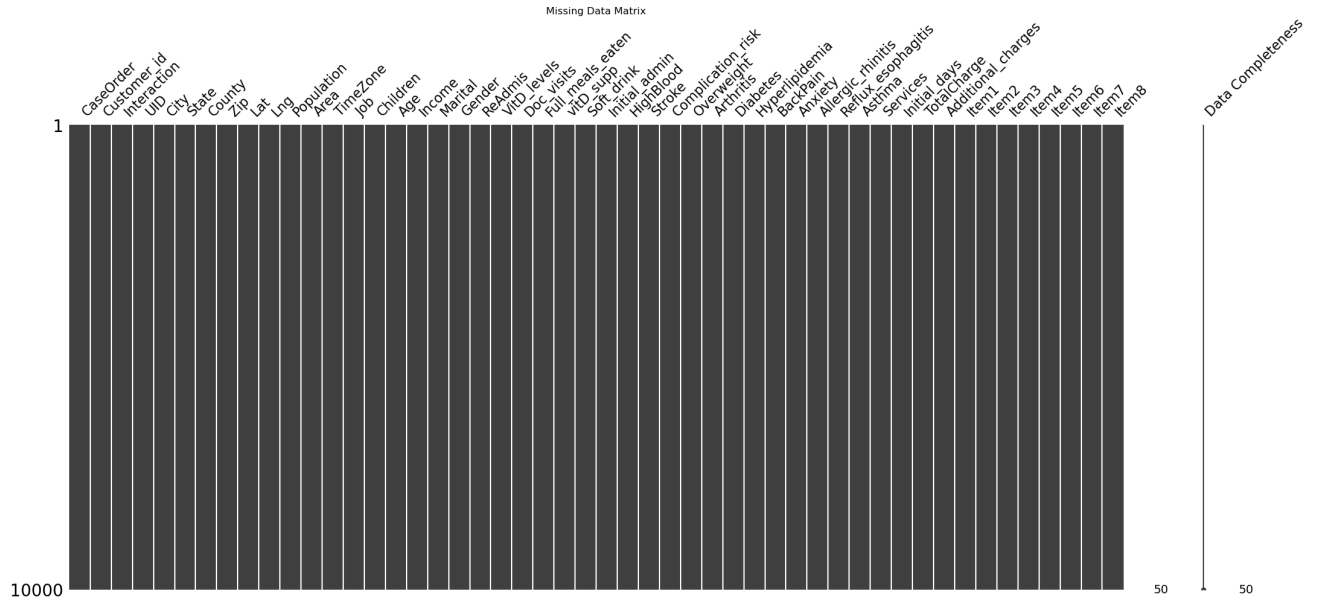
In [24]:
```
#Viewing Data to evaluate structure and types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CaseOrder           10000 non-null  int64  
 1   Customer_id         10000 non-null  object 
 2   Interaction         10000 non-null  object 
 3   UID                 10000 non-null  object 
 4   City                10000 non-null  object 
 5   State               10000 non-null  object 
 6   County              10000 non-null  object 
 7   Zip                 10000 non-null  int64  
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64  
 11  Area                10000 non-null  object 
 12  TimeZone            10000 non-null  object 
 13  Job                 10000 non-null  object 
 14  Children            10000 non-null  int64  
 15  Age                 10000 non-null  int64  
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object 
 18  Gender              10000 non-null  object 
 19  ReAdmis             10000 non-null  object 
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64  
 22  Full_meals_eaten    10000 non-null  int64  
 23  vitD_supp           10000 non-null  int64  
 24  Soft_drink          10000 non-null  object 
 25  Initial_admin       10000 non-null  object 
 26  HighBlood           10000 non-null  object 
 27  Stroke              10000 non-null  object 
 28  Complication_risk   10000 non-null  object 
 29  Overweight          10000 non-null  object 
 30  Arthritis           10000 non-null  object 
 31  Diabetes            10000 non-null  object 
 32  Hyperlipidemia      10000 non-null  object 
 33  BackPain            10000 non-null  object 
 34  Anxiety             10000 non-null  object 
 35  Allergic_rhinitis   10000 non-null  object 
 36  Reflux_esophagitis  10000 non-null  object 
 37  Asthma              10000 non-null  object 
 38  Services            10000 non-null  object 
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64  
 43  Item2               10000 non-null  int64  
 44  Item3               10000 non-null  int64  
 45  Item4               10000 non-null  int64  
 46  Item5               10000 non-null  int64  
 47  Item6               10000 non-null  int64  
 48  Item7               10000 non-null  int64  
 49  Item8               10000 non-null  int64  
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

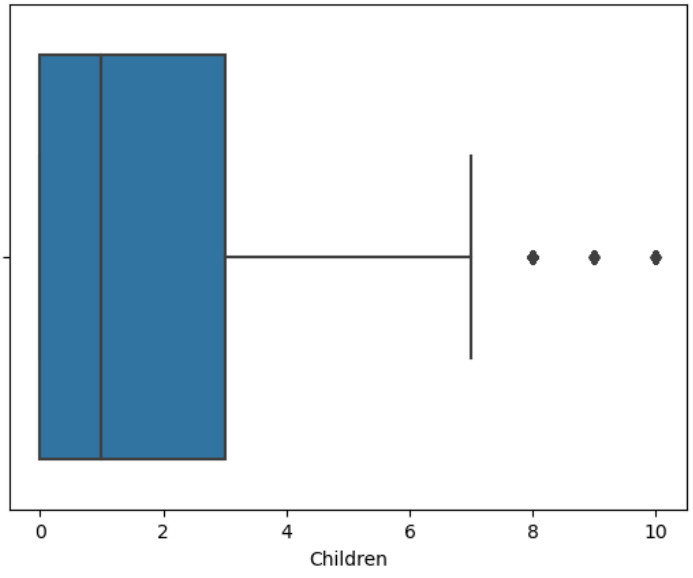
In [25]:
```
#Detect null values
print(df.isnull().sum())
```

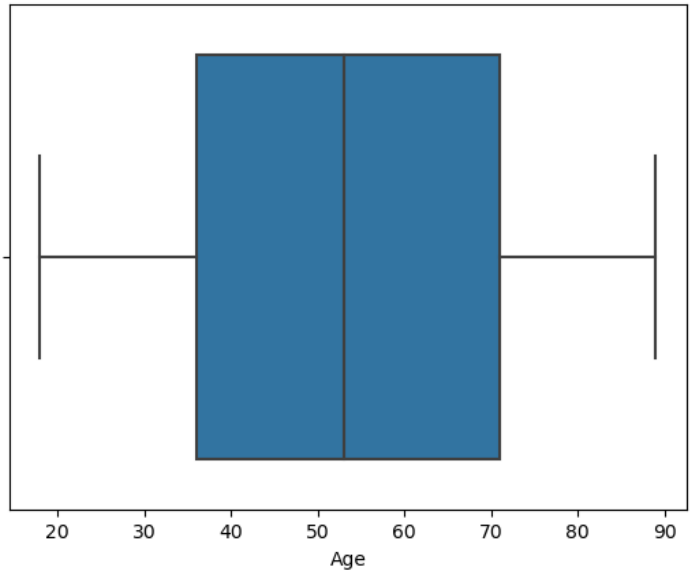
```
CaseOrder             0
Customer_id           0
Interaction           0
UID                   0
City                  0
State                 0
County                0
Zip                   0
Lat                   0
Lng                   0
Population            0
Area                  0
TimeZone              0
Job                   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
TotalCharge           0
Additional_charges    0
Item1                 0
Item2                 0
Item3                 0
Item4                 0
Item5                 0
Item6                 0
Item7                 0
Item8                 0
dtype: int64
```

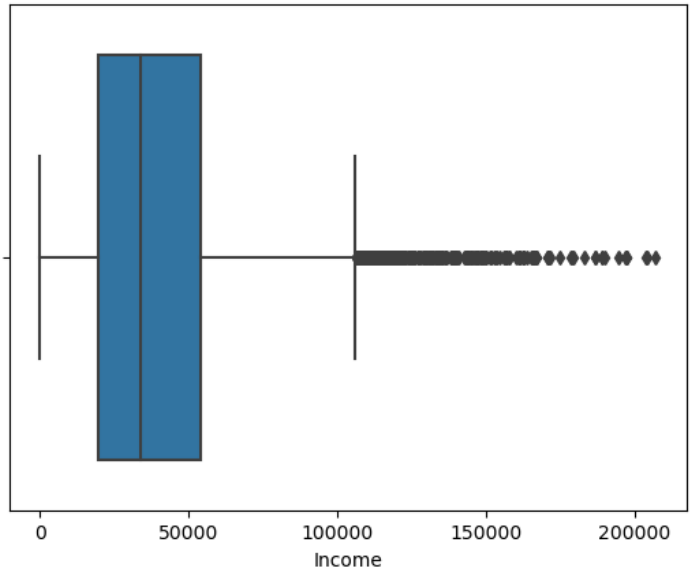
In [26]:
```
#Detect missing data
msno.matrix(df, labels=True)
plt.title('Missing Data Matrix')
plt.show()
```

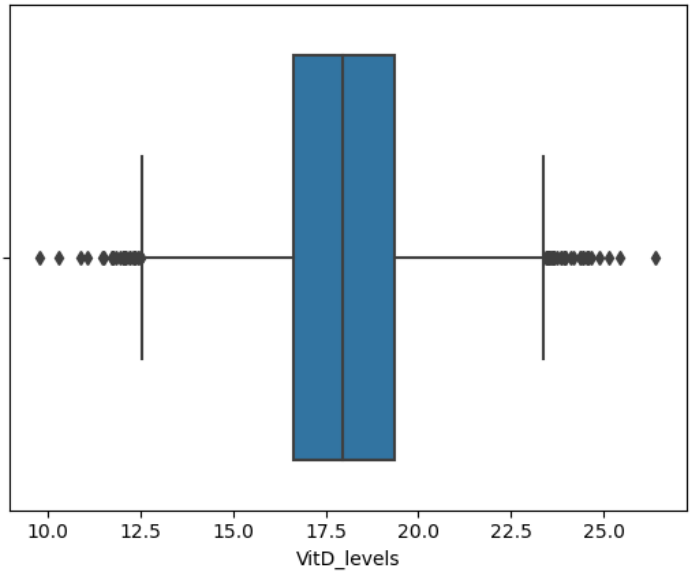


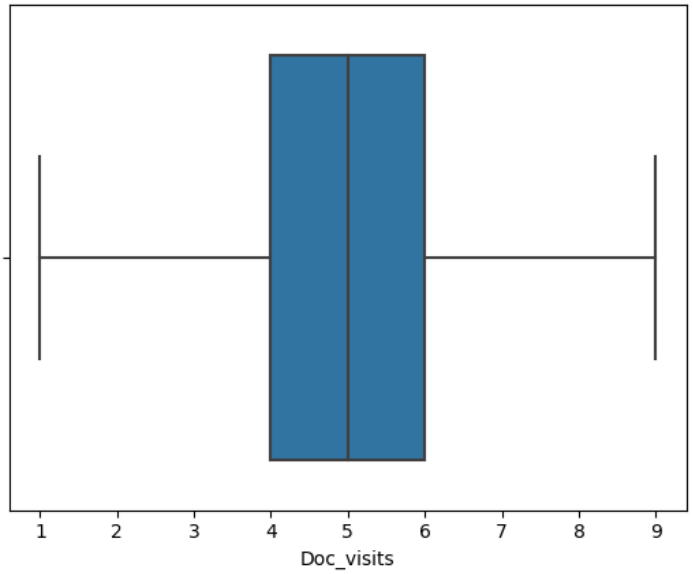
In [27]:
```
#Detection of outliers for quantitative values
boxplot=sns.boxplot(x='Children',data=df)
plt.show()
boxplot=sns.boxplot(x='Age',data=df)
plt.show()
boxplot=sns.boxplot(x='Income',data=df)
plt.show()
boxplot=sns.boxplot(x='VitD_levels',data=df)
plt.show()
boxplot=sns.boxplot(x='Doc_visits',data=df)
plt.show()
boxplot=sns.boxplot(x='Full_meals_eaten',data=df)
plt.show()
boxplot=sns.boxplot(x='vitD_supp',data=df)
plt.show()
boxplot=sns.boxplot(x='TotalCharge',data=df)
plt.show()
boxplot=sns.boxplot(x='Additional_charges',data=df)
plt.show()
boxplot=sns.boxplot(x='Item1',data=df)
plt.show()
boxplot=sns.boxplot(x='Item2',data=df)
plt.show()
boxplot=sns.boxplot(x='Item3',data=df)
plt.show()
boxplot=sns.boxplot(x='Item4',data=df)
plt.show()
boxplot=sns.boxplot(x='Item5',data=df)
plt.show()
boxplot=sns.boxplot(x='Item6',data=df)
plt.show()
boxplot=sns.boxplot(x='Item7',data=df)
plt.show()
boxplot=sns.boxplot(x='Item8',data=df)
plt.show()

#Visualizing distribution shapes
print('Children Original: ')
plt.hist(df['Children'])
plt.show()
print('Age Original: ')
plt.hist(df['Age'])
plt.show()
print('Income Original: ')
plt.hist(df['Income'])
plt.show()
print('Overweight Original: ')
plt.hist(df['Overweight'])
plt.show()
print('Anxiety Original: ')
plt.hist(df['Anxiety'])
plt.show()
print('Initial_days Original: ')
plt.hist(df['Initial_days'])
plt.show()
```

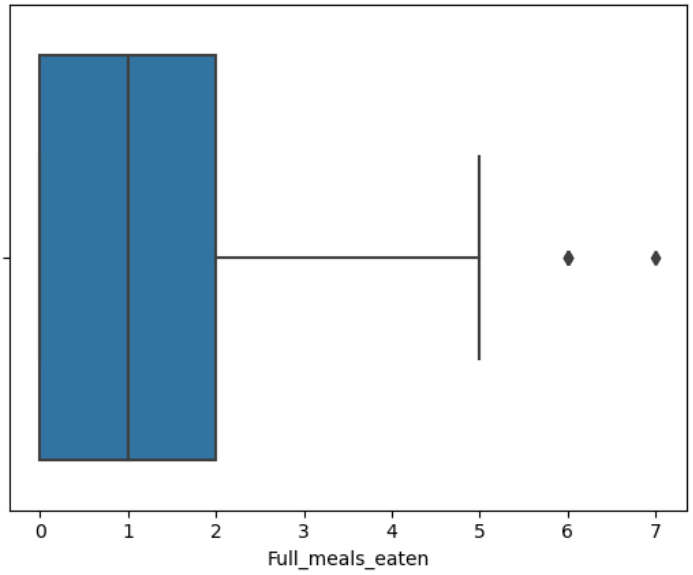


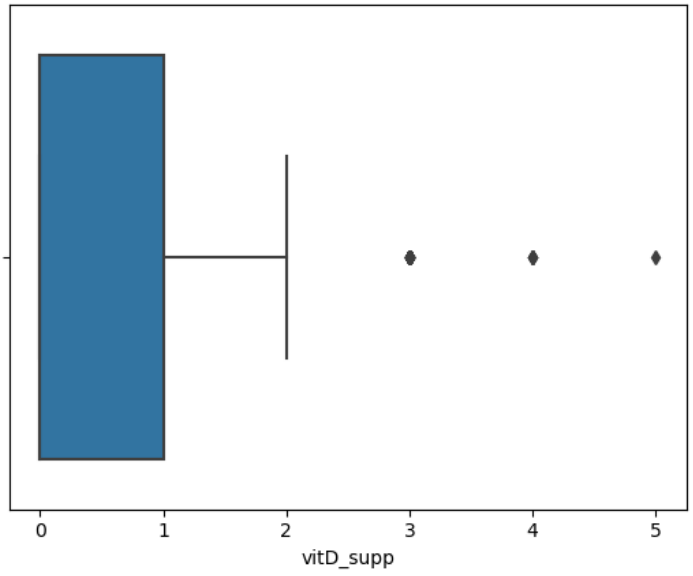

20
30
40
50
60
70
80
90
Age

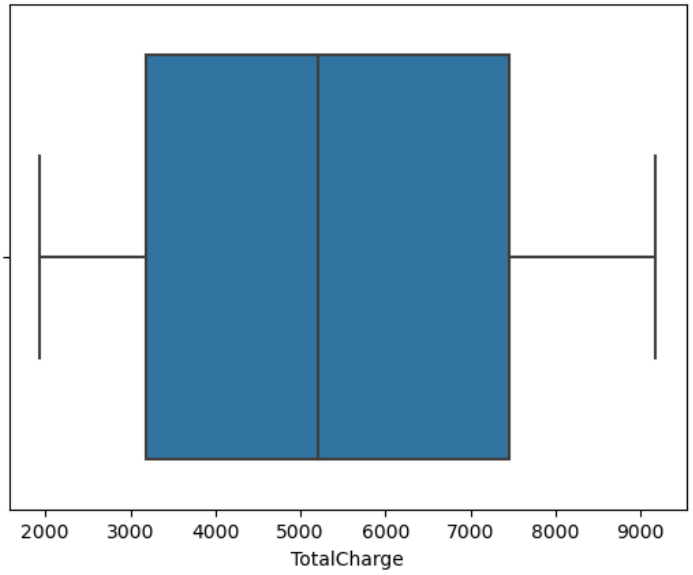

0
50000
100000
150000
200000
Income

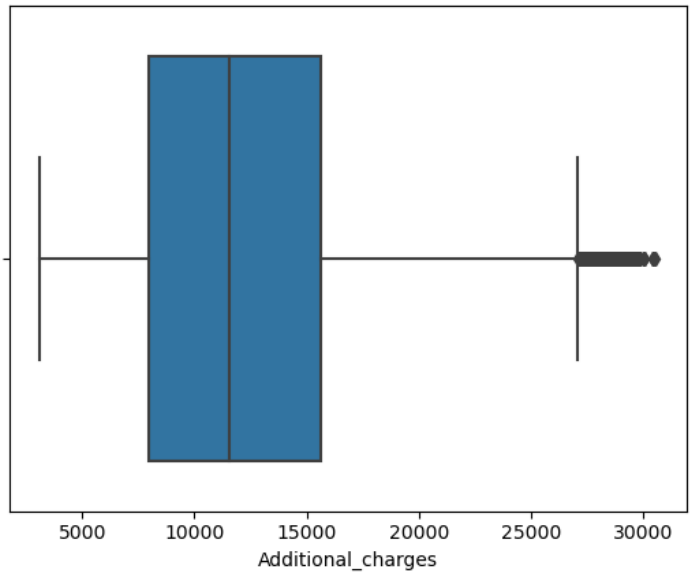

10.0
12.5
15.0
17.5
20.0
22.5
25.0
VitD_levels

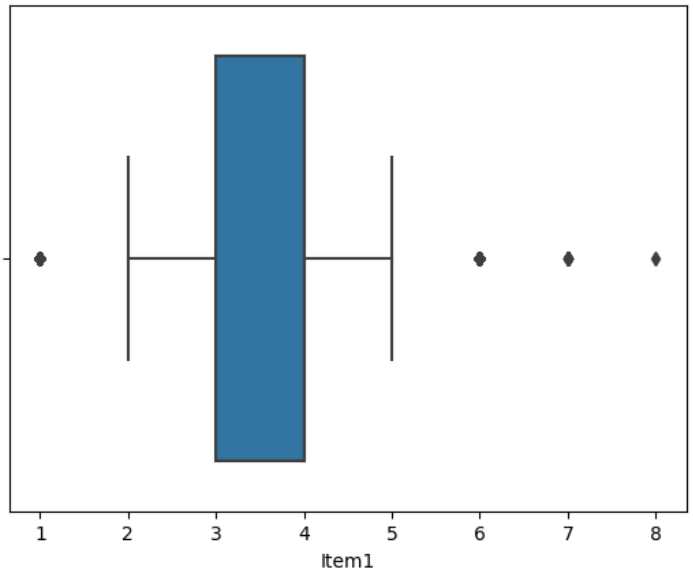

1
2
3
4
5
6
7
8
9
Doc_visits

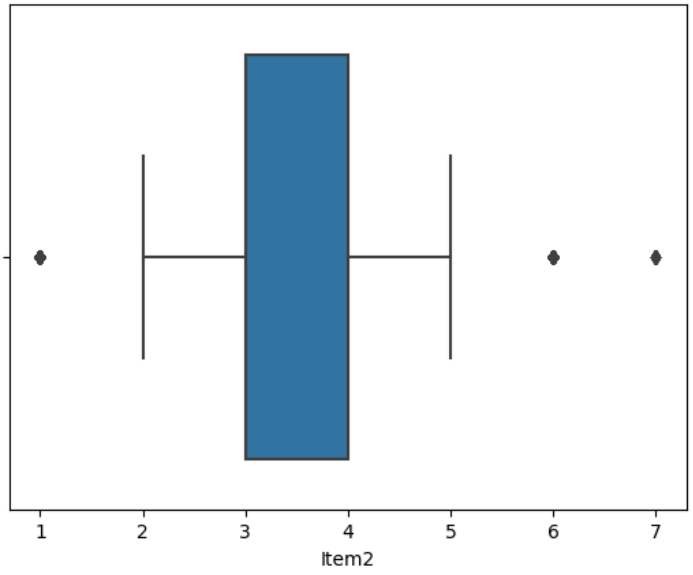

0
1
2
3
4
5
6
7
Full_meals_eaten

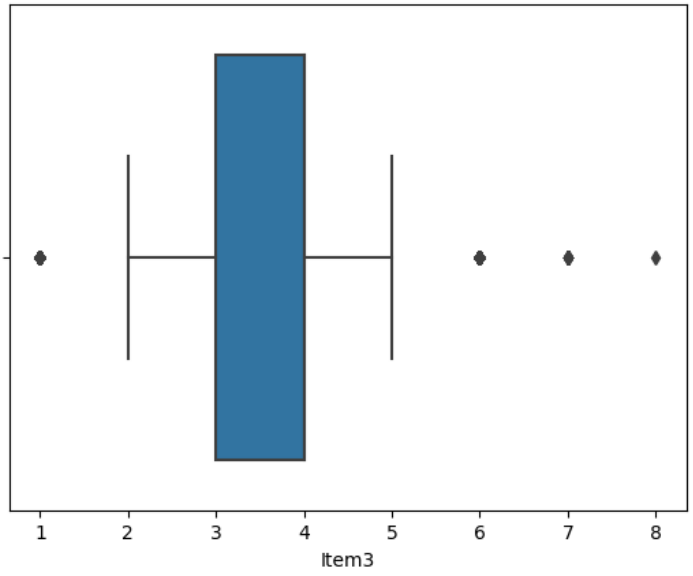

0
1
2
3
4
5
vitD_supp

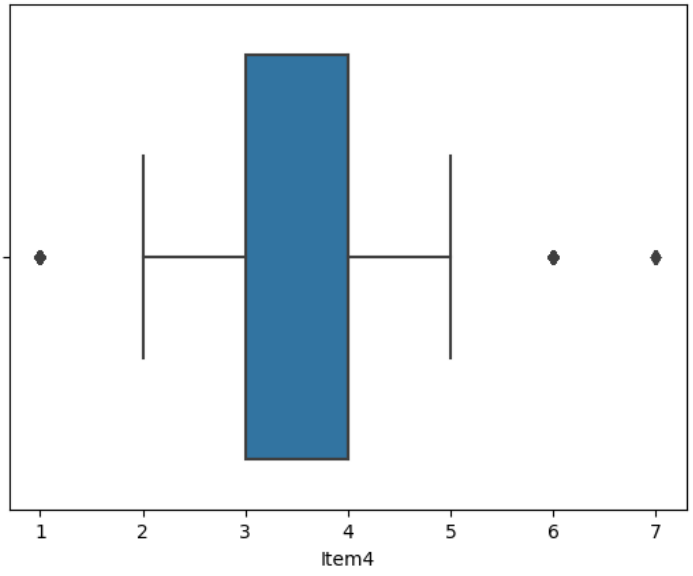

2000
3000
4000
5000
6000
7000
8000
9000
TotalCharge

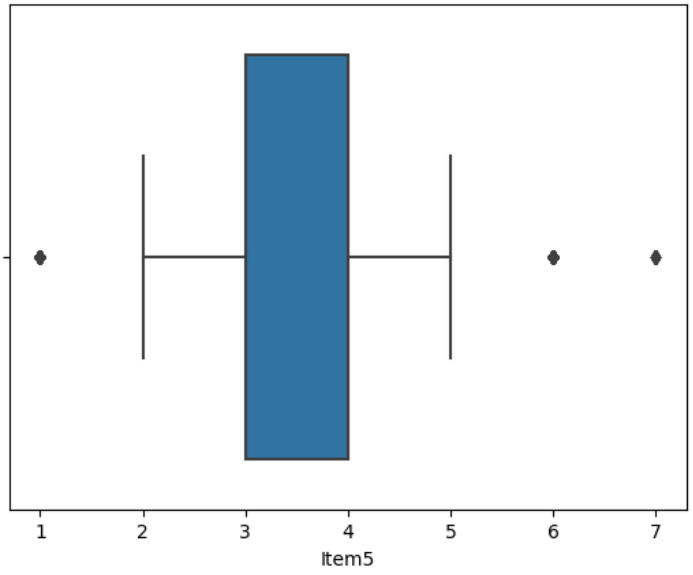

5000
10000
15000
20000
25000
30000
Additional_charges

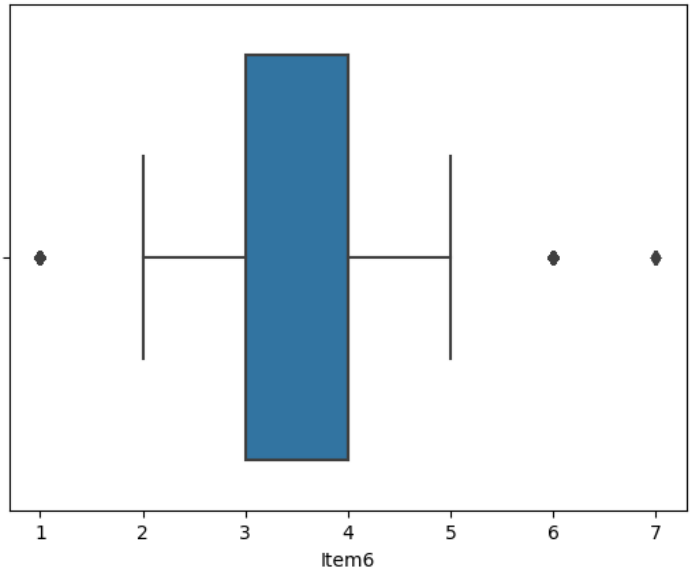

1
2
3
4
5
6
7
8
Item1

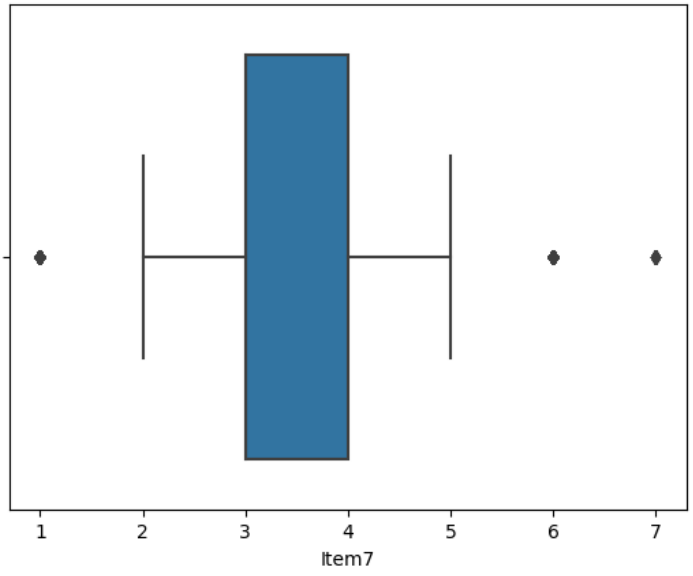

1
2
3
4
5
6
7
Item2

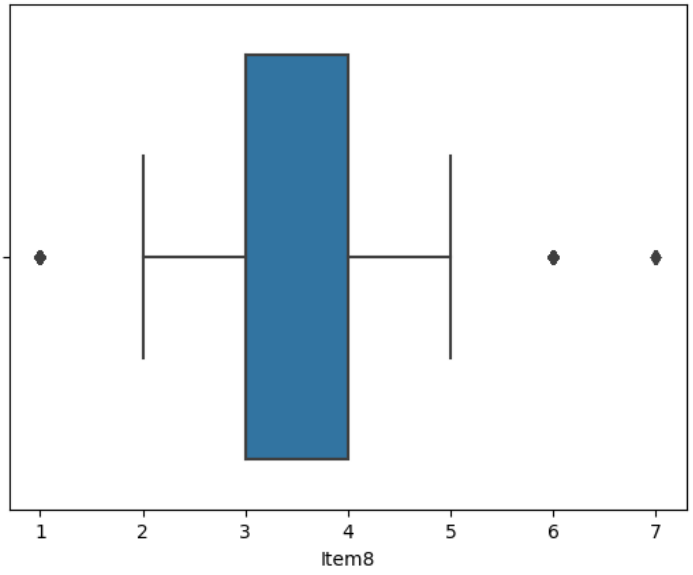

1
2
3
4
5
6
7
8
Item3


1
2
3
4
5
6
7
Item4

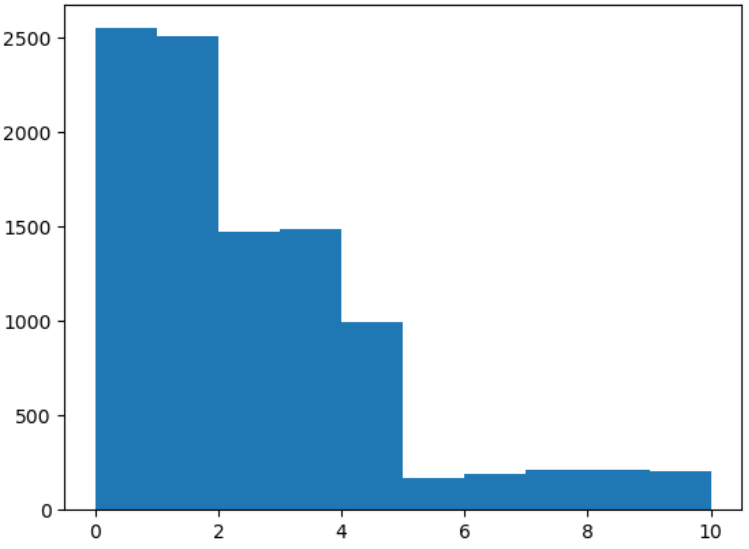

1
2
3
4
5
6
7
Item5


1
2
3
4
5
6
7
Item6

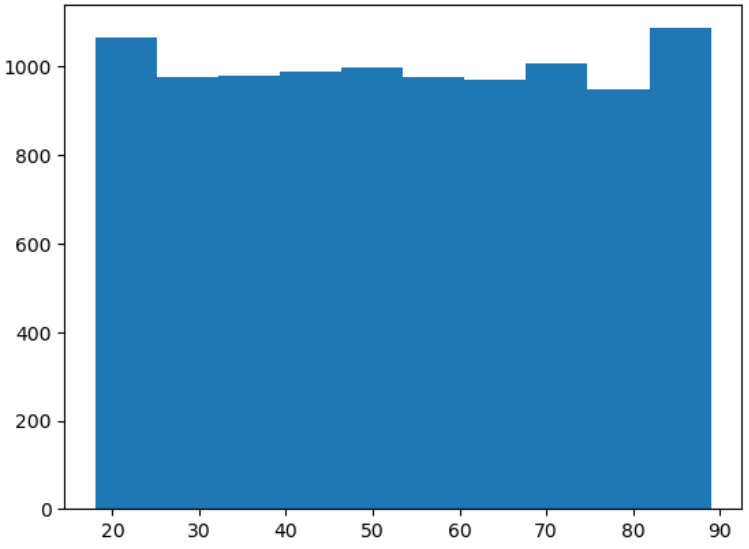

1
2
3
4
5
6
7
Item7



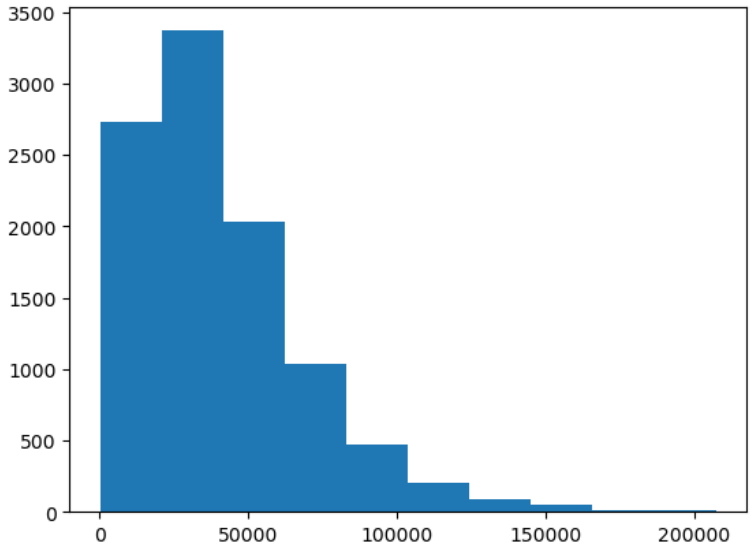

```
Children Original:
```



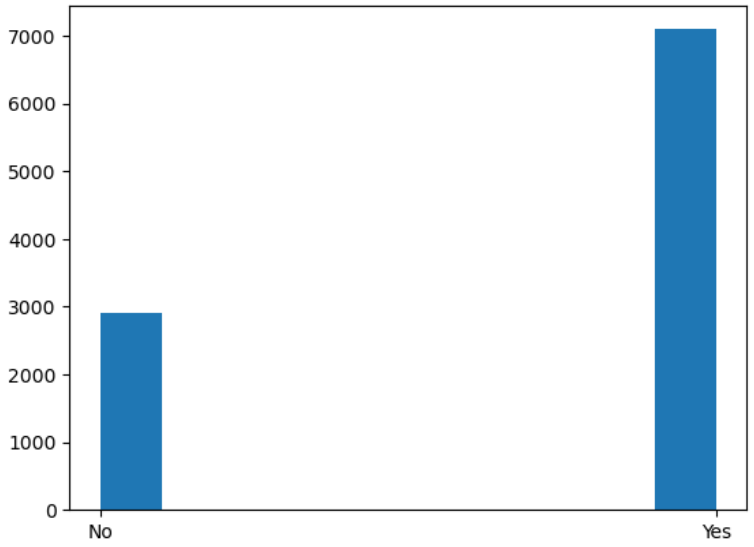

```
Age Original:
```



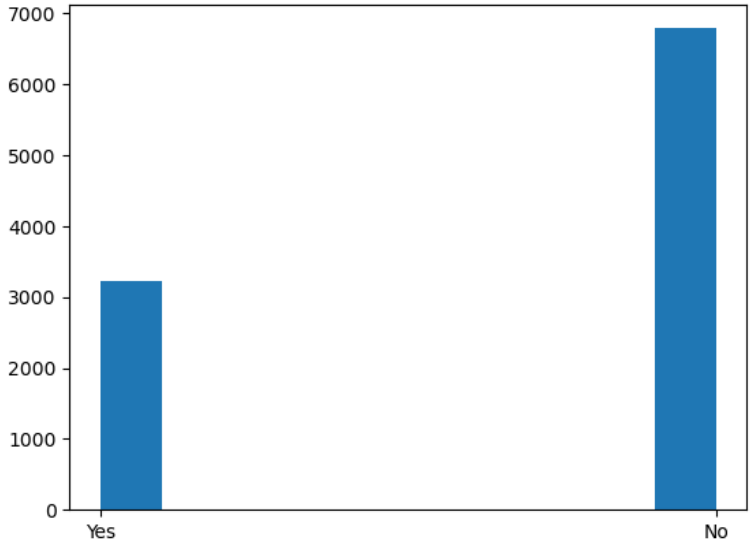

```
Income Original:
```



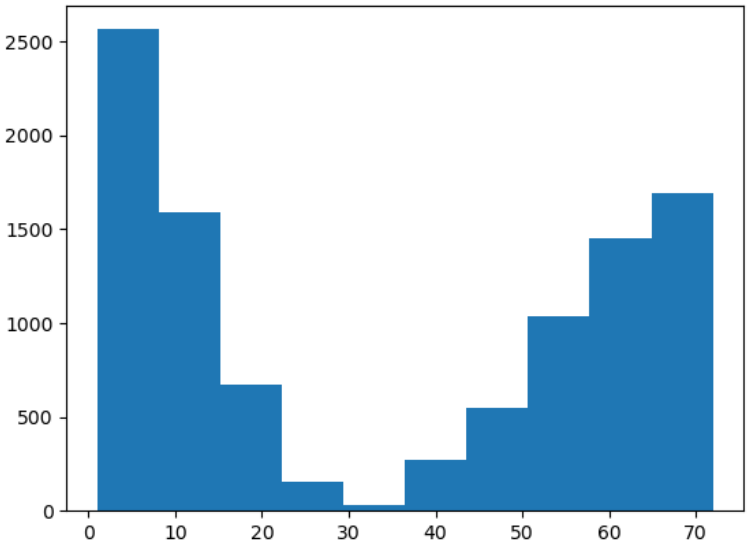

```
Overweight Original:
```




```
Anxiety Original:
```




```
Initial_days Original:
```


2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70

In [28]:
```
#Data Wrangling; turn categorical values into quantitative data
df['ReAdmis_numeric'] = df['ReAdmis']
dict_ReAdmis = {"ReAdmis_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_ReAdmis, inplace=True)

df['Soft_drink_numeric'] = df['Soft_drink']
dict_Soft_drink = {"Soft_drink_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_Soft_drink, inplace=True)

df['HighBlood_numeric'] = df['HighBlood']
dict_HighBlood = {"HighBlood_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_HighBlood, inplace=True)

df['Stroke_numeric'] = df['Stroke']
dict_stroke = {"Stroke_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_stroke, inplace=True)

df['Arthritis_numeric'] = df['Arthritis']
dict_arthritis = {"Arthritis_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_arthritis, inplace=True)

df['Diabetes_numeric'] = df['Diabetes']
dict_diabetes = {"Diabetes_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_diabetes, inplace=True)

df['Hyperlipidemia_numeric'] = df['Hyperlipidemia']
dict_hyperlipidemia = {"Hyperlipidemia_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_hyperlipidemia, inplace=True)

df['BackPain_numeric'] = df['BackPain']
dict_backpain = {"BackPain_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_backpain, inplace=True)

df['Allergic_rhinitis_numeric'] = df['Allergic_rhinitis']
dict_allergies = {"Allergic_rhinitis_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_allergies, inplace=True)

df['Reflux_esophagitis_numeric'] = df['Reflux_esophagitis']
dict_reflux = {"Reflux_esophagitis_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_reflux, inplace=True)

df['Asthma_numeric'] = df['Asthma']
dict_asthma = {"Asthma_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_asthma, inplace=True)

df['Overweight_numeric'] = df['Overweight']
dict_Overweight = {"Overweight_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_Overweight, inplace=True)

df['Anxiety_numeric'] = df['Anxiety']
dict_Anxiety = {"Anxiety_numeric": {"No": 0, "Yes": 1}}
df.replace(dict_Anxiety, inplace=True)

df = pd.get_dummies(df, columns=["Marital", "Services", "Gender", "Initial_admin", "Complication_risk"])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 76 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   CaseOrder                             10000 non-null  int64  
 1   Customer_id                           10000 non-null  object 
 2   Interaction                           10000 non-null  object 
 3   UID                                   10000 non-null  object 
 4   City                                  10000 non-null  object 
 5   State                                 10000 non-null  object 
 6   County                                10000 non-null  object 
 7   Zip                                   10000 non-null  int64  
 8   Lat                                   10000 non-null  float64
 9   Lng                                   10000 non-null  float64
 10  Population                            10000 non-null  int64  
 11  Area                                  10000 non-null  object 
 12  TimeZone                              10000 non-null  object 
 13  Job                                   10000 non-null  object 
 14  Children                              10000 non-null  int64  
 15  Age                                   10000 non-null  int64  
 16  Income                                10000 non-null  float64
 17  ReAdmis                               10000 non-null  object 
 18  VitD_levels                           10000 non-null  float64
 19  Doc_visits                            10000 non-null  int64  
 20  Full_meals_eaten                      10000 non-null  int64  
 21  vitD_supp                             10000 non-null  int64  
 22  Soft_drink                            10000 non-null  object 
 23  HighBlood                             10000 non-null  object 
 24  Stroke                                10000 non-null  object 
 25  Overweight                            10000 non-null  object 
 26  Arthritis                             10000 non-null  object 
 27  Diabetes                              10000 non-null  object 
 28  Hyperlipidemia                        10000 non-null  object 
 29  BackPain                              10000 non-null  object 
 30  Anxiety                               10000 non-null  object 
 31  Allergic_rhinitis                     10000 non-null  object 
 32  Reflux_esophagitis                    10000 non-null  object 
 33  Asthma                                10000 non-null  object 
 34  Initial_days                          10000 non-null  float64
 35  TotalCharge                           10000 non-null  float64
 36  Additional_charges                    10000 non-null  float64
 37  Item1                                 10000 non-null  int64  
 38  Item2                                 10000 non-null  int64  
 39  Item3                                 10000 non-null  int64  
 40  Item4                                 10000 non-null  int64  
 41  Item5                                 10000 non-null  int64  
 42  Item6                                 10000 non-null  int64  
 43  Item7                                 10000 non-null  int64  
 44  Item8                                 10000 non-null  int64  
 45  ReAdmis_numeric                       10000 non-null  int64  
 46  Soft_drink_numeric                    10000 non-null  int64  
 47  HighBlood_numeric                     10000 non-null  int64  
 48  Stroke_numeric                        10000 non-null  int64  
 49  Arthritis_numeric                     10000 non-null  int64  
 50  Diabetes_numeric                      10000 non-null  int64  
 51  Hyperlipidemia_numeric                10000 non-null  int64  
 52  BackPain_numeric                      10000 non-null  int64  
 53  Allergic_rhinitis_numeric             10000 non-null  int64  
 54  Reflux_esophagitis_numeric            10000 non-null  int64  
 55  Asthma_numeric                        10000 non-null  int64  
 56  Overweight_numeric                    10000 non-null  int64  
 57  Anxiety_numeric                       10000 non-null  int64  
 58  Marital_Divorced                      10000 non-null  uint8  
 59  Marital_Married                       10000 non-null  uint8  
 60  Marital_Never Married                 10000 non-null  uint8  
 61  Marital_Separated                     10000 non-null  uint8  
 62  Marital_Widowed                       10000 non-null  uint8  
 63  Services_Blood Work                   10000 non-null  uint8  
 64  Services_CT Scan                      10000 non-null  uint8  
 65  Services_Intravenous                  10000 non-null  uint8  
 66  Services_MRI                          10000 non-null  uint8  
 67  Gender_Female                         10000 non-null  uint8  
 68  Gender_Male                           10000 non-null  uint8  
 69  Gender_Nonbinary                      10000 non-null  uint8  
 70  Initial_admin_Elective Admission      10000 non-null  uint8  
 71  Initial_admin_Emergency Admission     10000 non-null  uint8  
 72  Initial_admin_Observation Admission   10000 non-null  uint8  
 73  Complication_risk_High                10000 non-null  uint8  
 74  Complication_risk_Low                 10000 non-null  uint8  
 75  Complication_risk_Medium              10000 non-null  uint8  
dtypes: float64(7), int64(29), object(22), uint8(18)
memory usage: 4.6+ MB
```

In [29]:
```
#Renaming columns from pd.get_dummies
df = df.rename({'Initial_admin_Elective Admission': 'Initial_admin_Elective_Admission',
                'Initial_admin_Emergency Admission': 'Initial_admin_Emergency_Admission',
                'Initial_admin_Observation Admission': 'Initial_admin_Observation_Admission',
                'Marital_Never Married': 'Marital_Never_Married',
                'Services_Blood Work': 'Services_Blood_Work',
                'Services_CT Scan': 'Services_CT_Scan'}, axis ='columns')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 76 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   CaseOrder                             10000 non-null  int64  
 1   Customer_id                           10000 non-null  object 
 2   Interaction                           10000 non-null  object 
 3   UID                                   10000 non-null  object 
 4   City                                  10000 non-null  object 
 5   State                                 10000 non-null  object 
 6   County                                10000 non-null  object 
 7   Zip                                   10000 non-null  int64  
 8   Lat                                   10000 non-null  float64
 9   Lng                                   10000 non-null  float64
 10  Population                            10000 non-null  int64  
 11  Area                                  10000 non-null  object 
 12  TimeZone                              10000 non-null  object 
 13  Job                                   10000 non-null  object 
 14  Children                              10000 non-null  int64  
 15  Age                                   10000 non-null  int64  
 16  Income                                10000 non-null  float64
 17  ReAdmis                               10000 non-null  object 
 18  VitD_levels                           10000 non-null  float64
 19  Doc_visits                            10000 non-null  int64  
 20  Full_meals_eaten                      10000 non-null  int64  
 21  vitD_supp                             10000 non-null  int64  
 22  Soft_drink                            10000 non-null  object 
 23  HighBlood                             10000 non-null  object 
 24  Stroke                                10000 non-null  object 
 25  Overweight                            10000 non-null  object 
 26  Arthritis                             10000 non-null  object 
 27  Diabetes                              10000 non-null  object 
 28  Hyperlipidemia                        10000 non-null  object 
 29  BackPain                              10000 non-null  object 
 30  Anxiety                               10000 non-null  object 
 31  Allergic_rhinitis                     10000 non-null  object 
 32  Reflux_esophagitis                    10000 non-null  object 
 33  Asthma                                10000 non-null  object 
 34  Initial_days                          10000 non-null  float64
 35  TotalCharge                           10000 non-null  float64
 36  Additional_charges                    10000 non-null  float64
 37  Item1                                 10000 non-null  int64  
 38  Item2                                 10000 non-null  int64  
 39  Item3                                 10000 non-null  int64  
 40  Item4                                 10000 non-null  int64  
 41  Item5                                 10000 non-null  int64  
 42  Item6                                 10000 non-null  int64  
 43  Item7                                 10000 non-null  int64  
 44  Item8                                 10000 non-null  int64  
 45  ReAdmis_numeric                       10000 non-null  int64  
 46  Soft_drink_numeric                    10000 non-null  int64  
 47  HighBlood_numeric                     10000 non-null  int64  
 48  Stroke_numeric                        10000 non-null  int64  
 49  Arthritis_numeric                     10000 non-null  int64  
 50  Diabetes_numeric                      10000 non-null  int64  
 51  Hyperlipidemia_numeric                10000 non-null  int64  
 52  BackPain_numeric                      10000 non-null  int64  
 53  Allergic_rhinitis_numeric             10000 non-null  int64  
 54  Reflux_esophagitis_numeric            10000 non-null  int64  
 55  Asthma_numeric                        10000 non-null  int64  
 56  Overweight_numeric                    10000 non-null  int64  
 57  Anxiety_numeric                       10000 non-null  int64  
 58  Marital_Divorced                      10000 non-null  uint8  
 59  Marital_Married                       10000 non-null  uint8  
 60  Marital_Never_Married                 10000 non-null  uint8  
 61  Marital_Separated                     10000 non-null  uint8  
 62  Marital_Widowed                       10000 non-null  uint8  
 63  Services_Blood_Work                   10000 non-null  uint8  
 64  Services_CT_Scan                      10000 non-null  uint8  
 65  Services_Intravenous                  10000 non-null  uint8  
 66  Services_MRI                          10000 non-null  uint8  
 67  Gender_Female                         10000 non-null  uint8  
 68  Gender_Male                           10000 non-null  uint8  
 69  Gender_Nonbinary                      10000 non-null  uint8  
 70  Initial_admin_Elective_Admission      10000 non-null  uint8  
 71  Initial_admin_Emergency_Admission     10000 non-null  uint8  
 72  Initial_admin_Observation_Admission   10000 non-null  uint8  
 73  Complication_risk_High                10000 non-null  uint8  
 74  Complication_risk_Low                 10000 non-null  uint8  
 75  Complication_risk_Medium              10000 non-null  uint8  
dtypes: float64(7), int64(29), object(22), uint8(18)
memory usage: 4.6+ MB
```

In [30]:
```
##Univariate Stats Dataframe
def unistats(df):
    output_df = pd.DataFrame(columns=['Count', 'Missing', 'Unique', 'Dtype', 'Numeric', 'Mean', 'Mode', 'Min', 'Median', 'Max

    for col in df:
        if pd.api.types.is_numeric_dtype(df[col]):
            output_df.loc[col] = [df[col].count(), df[col].isnull().sum(), df[col].nunique(), df[col].dtype, pd.api.types.is_
        else:
            output_df.loc[col] = [df[col].count(), df[col].isnull().sum(), df[col].nunique(), df[col].dtype, pd.api.types.is_
    return output_df.sort_values(by=['Numeric', 'Skew', 'Unique'], ascending=False)

df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'State', 'County', 'Job', 'Zip', 'TimeZone', 'Lat', 'Lng',
print(unistats(df))
```

	Count	Missing	Unique	Dtype	Numeric
Gender_Nonbinary	10000	0	2	uint8	True
Services_MRI	10000	0	2	uint8	True
Services_CT_Scan	10000	0	2	uint8	True
Population	10000	0	5951	int64	True
vitD_supp	10000	0	6	int64	True
Marital_Never_Married	10000	0	2	uint8	True
Marital_Separated	10000	0	2	uint8	True
Stroke_numeric	10000	0	2	int64	True
Marital_Married	10000	0	2	uint8	True
Marital_Widowed	10000	0	2	uint8	True
Children	10000	0	11	int64	True
Income	10000	0	9993	float64	True
Complication_risk_Low	10000	0	2	uint8	True
Initial_admin_Observation_Admission	10000	0	2	uint8	True
Initial_admin_Elective_Admission	10000	0	2	uint8	True
Soft_drink_numeric	10000	0	2	int64	True
Diabetes_numeric	10000	0	2	int64	True
Full_meals_eaten	10000	0	8	int64	True
Asthma_numeric	10000	0	2	int64	True
Additional_charges	10000	0	9418	float64	True
Services_Intravenous	10000	0	2	uint8	True
Anxiety_numeric	10000	0	2	int64	True
Complication_risk_High	10000	0	2	uint8	True
Hyperlipidemia_numeric	10000	0	2	int64	True
Arthritis_numeric	10000	0	2	int64	True
ReAdmis_numeric	10000	0	2	int64	True
Allergic_rhinitis_numeric	10000	0	2	int64	True
HighBlood_numeric	10000	0	2	int64	True
BackPain_numeric	10000	0	2	int64	True
Reflux_esophagitis_numeric	10000	0	2	int64	True
Complication_risk_Medium	10000	0	2	uint8	True
Gender_Male	10000	0	2	uint8	True
Initial_days	10000	0	9997	float64	True
TotalCharge	10000	0	9997	float64	True
VitD_levels	10000	0	9976	float64	True
Age	10000	0	72	int64	True
Doc_visits	10000	0	9	int64	True
Initial_admin_Emergency_Admission	10000	0	2	uint8	True
Services_Blood_Work	10000	0	2	uint8	True
Overweight_numeric	10000	0	2	int64	True

	Mean	Mode	Min
Gender_Nonbinary	0.021400	0.00000	0.000000
Services_MRI	0.038000	0.00000	0.000000
Services_CT_Scan	0.122500	0.00000	0.000000
Population	9965.253800	0.00000	0.000000
vitD_supp	0.398900	0.00000	0.000000
Marital_Never_Married	0.198400	0.00000	0.000000
Marital_Separated	0.198700	0.00000	0.000000
Stroke_numeric	0.199300	0.00000	0.000000
Marital_Married	0.202300	0.00000	0.000000
Marital_Widowed	0.204500	0.00000	0.000000
Children	2.097200	0.00000	0.000000
Income	40490.495160	14572.40000	154.080000
Complication_risk_Low	0.212500	0.00000	0.000000
Initial_admin_Observation_Admission	0.243600	0.00000	0.000000
Initial_admin_Elective_Admission	0.250400	0.00000	0.000000
Soft_drink_numeric	0.257500	0.00000	0.000000
Diabetes_numeric	0.273800	0.00000	0.000000
Full_meals_eaten	1.001400	0.00000	0.000000
Asthma_numeric	0.289300	0.00000	0.000000
Additional_charges	12934.528587	3883.66416	3125.703000
Services_Intravenous	0.313000	0.00000	0.000000
Anxiety_numeric	0.321500	0.00000	0.000000
Complication_risk_High	0.335800	0.00000	0.000000
Hyperlipidemia_numeric	0.337200	0.00000	0.000000
Arthritis_numeric	0.357400	0.00000	0.000000
ReAdmis_numeric	0.366900	0.00000	0.000000
Allergic_rhinitis_numeric	0.394100	0.00000	0.000000
HighBlood_numeric	0.409000	0.00000	0.000000
BackPain_numeric	0.411400	0.00000	0.000000
Reflux_esophagitis_numeric	0.413500	0.00000	0.000000
Complication_risk_Medium	0.451700	0.00000	0.000000
Gender_Male	0.476800	0.00000	0.000000
Initial_days	34.455299	63.54432	1.001981
TotalCharge	5312.172769	7555.45200	1938.312067
VitD_levels	17.964262	15.26009	9.806483
Age	53.511700	47.00000	18.000000
Doc_visits	5.012200	5.00000	1.000000
Initial_admin_Emergency_Admission	0.506000	1.00000	0.000000
Services_Blood_Work	0.526500	1.00000	0.000000
Overweight_numeric	0.709400	1.00000	0.000000

	Median	Max	\
Gender_Nonbinary	0.000000	1.000000	
Services_MRI	0.000000	1.000000	
Services_CT_Scan	0.000000	1.000000	
Population	2769.000000	122814.000000	
vitD_supp	0.000000	5.000000	
Marital_Never_Married	0.000000	1.000000	
Marital_Separated	0.000000	1.000000	
Stroke_numeric	0.000000	1.000000	
Marital_Married	0.000000	1.000000	
Marital_Widowed	0.000000	1.000000	
Children	1.000000	10.000000	
Income	33768.420000	207249.100000	
Complication_risk_Low	0.000000	1.000000	
Initial_admin_Observation_Admission	0.000000	1.000000	
Initial_admin_Elective_Admission	0.000000	1.000000	
Soft_drink_numeric	0.000000	1.000000	
Diabetes_numeric	0.000000	1.000000	
Full_meals_eaten	1.000000	7.000000	
Asthma_numeric	0.000000	1.000000	
Additional_charges	11573.977735	30566.070000	
Services_Intravenous	0.000000	1.000000	
Anxiety_numeric	0.000000	1.000000	
Complication_risk_High	0.000000	1.000000	
Hyperlipidemia_numeric	0.000000	1.000000	
Arthritis_numeric	0.000000	1.000000	
ReAdmis_numeric	0.000000	1.000000	
Allergic_rhinitis_numeric	0.000000	1.000000	
HighBlood_numeric	0.000000	1.000000	
BackPain_numeric	0.000000	1.000000	
Reflux_esophagitis_numeric	0.000000	1.000000	
Complication_risk_Medium	0.000000	1.000000	
Gender_Male	0.000000	1.000000	
Initial_days	35.836244	71.981490	
TotalCharge	5213.952000	9180.728000	
VitD_levels	17.951122	26.394449	
Age	53.000000	89.000000	
Doc_visits	5.000000	9.000000	
Initial_admin_Emergency_Admission	1.000000	1.000000	
Services_Blood_Work	1.000000	1.000000	
Overweight_numeric	1.000000	1.000000	

	Std	Skew	Kurt
Gender_Nonbinary	0.144721	6.615434	41.772323
Services_MRI	0.191206	4.833456	21.366572
Services_CT_Scan	0.327879	2.303141	3.305119
Population	14824.758614	2.229959	5.880913
vitD_supp	0.628505	1.550205	2.330763
Marital_Never_Married	0.398815	1.512784	0.288572
Marital_Separated	0.399042	1.510420	0.281425
Stroke_numeric	0.399494	1.505705	0.267202
Marital_Married	0.401735	1.482369	0.197456
Marital_Widowed	0.403356	1.465500	0.147720
Children	2.163659	1.448013	2.076321
Income	28521.153293	1.405899	2.745690
Complication_risk_Low	0.409097	1.405815	-0.023688
Initial_admin_Observation_Admission	0.429276	1.194810	-0.572544
Initial_admin_Elective_Admission	0.433265	1.152412	-0.672081
Soft_drink_numeric	0.437279	1.109354	-0.769488
Diabetes_numeric	0.445930	1.014712	-0.970553
Full_meals_eaten	1.008117	1.009461	1.042727
Asthma_numeric	0.453460	0.929485	-1.136285
Additional_charges	6542.601544	0.831842	-0.142684
Services_Intravenous	0.463738	0.806652	-1.349583
Anxiety_numeric	0.467076	0.764483	-1.415849
Complication_risk_High	0.472293	0.695470	-1.516625
Hyperlipidemia_numeric	0.472777	0.688834	-1.525813
Arthritis_numeric	0.479258	0.595206	-1.646059
ReAdmis_numeric	0.481983	0.552412	-1.695180
Allergic_rhinitis_numeric	0.488681	0.433498	-1.812442
HighBlood_numeric	0.491674	0.370238	-1.863296
BackPain_numeric	0.492112	0.360153	-1.870664
Reflux_esophagitis_numeric	0.492486	0.351350	-1.876929
Complication_risk_Medium	0.497687	0.194137	-1.962703
Gender_Male	0.499486	0.092914	-1.991765
Initial_days	26.309341	0.070286	-1.754525
TotalCharge	2180.393838	0.069661	-1.668267
VitD_levels	2.017231	0.032435	-0.022112
Age	20.638538	0.005117	-1.189527
Doc_visits	1.045734	-0.018563	0.025999
Initial_admin_Emergency_Admission	0.499989	-0.024005	-1.999824
Services_Blood_Work	0.499322	-0.106165	-1.989127
Overweight_numeric	0.454062	-0.922526	-1.149176

In [43]:
```
#C4. Cleaned Dataset:
# Provide a copy of the cleaned Data Set
df.to_csv(r'C:\Users\mmorg\WGU\D209\Cleaned209data.csv')
```

```
Part IV: Analysis
```

In [ ]:
```
D1. Split Data into Training and Test Data Sets and Provide the File(s)
```

In [32]:
```
#Set predictor variables & target variable
X = df[['Initial_days', 'Services_CT_Scan', 'Children', 'Services_Intravenous', 'Population', 'Initial_admin_Emergency_Admissi
y = df["ReAdmis_numeric"]
```

In [33]:
```
SEED = 1

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.7,test_size=0.3,
                                                    random_state=15,stratify=y)
```

In [34]:
```
# Instantiate KNN model
knn = KNeighborsClassifier(n_neighbors = 1)

# Fit data to KNN model
knn.fit(X_train, y_train)

# Predict outcomes from test set
y_pred = knn.predict(X_test)
```

In [35]:
```
#export training and test set to csv files
X_train.to_csv(r'C:\Users\mmorg\WGU\D209\X_train.csv')
X_test.to_csv(r'C:\Users\mmorg\WGU\D209\X_test.csv')
y_train.to_csv(r'C:\Users\mmorg\WGU\D209\y_train.csv')
y_test.to_csv(r'C:\Users\mmorg\WGU\D209\y_test.csv')
```

```
D2. Describe Analysis Technique Used
k-NN was used to predict patients who are at risk of re-admission into the hospital. I used GridSearchCV to analyze the
dataset and determine the n_neighbors to use, which was 1. I also used SelectKBest to choose the best features to use in
future analysis with a tolerance of p-values < 0.05.
```

In [ ]:
```
D3. Provide the code used to perform the classficiation analysis from part D2.
```

In [36]:
```
# Import GridSearchCV for cross validation of model
from sklearn.model_selection import GridSearchCV

# Set up parameters grid
param_grid = {'n_neighbors': np.arange(1, 50)}

# Re-intantiate KNN for cross validation
knn = KNeighborsClassifier()

# Instantiate GridSearch cross validation
knn_cv = GridSearchCV(knn , param_grid, cv=5)

# Fit model to
knn_cv.fit(X_train, y_train)

# Print best parameters
print('Best parameters for this KNN model: {}'.format(knn_cv.best_params_))
```

```
Best parameters for this KNN model: {'n_neighbors': 1}
```

In [37]:
```
#SelectKBest Code
#assign values for x for all predictor variables
#assign values for y for the dependent variable

print(X.shape)
print(y.shape)

feature_names = X.columns
#initialize the class and call fit_transform
from sklearn.feature_selection import SelectKBest, f_classif
skbest = SelectKBest(score_func = f_classif, k='all')
X_new = skbest.fit_transform(X,y)
X_new.shape

##Finding p-values to select statistically significant features
p_values = pd.DataFrame({'Feature': X.columns,
'p_value':skbest.pvalues_}).sort_values('p_value')
p_values[p_values['p_value']<.05]

features_to_keep = p_values['Feature'][p_values['p_value']<.05]
# Print the name of the selected features
features_to_keep
```

```
(10000, 6)
(10000,)
```

Out[37]:
```
0                        Initial_days
1                    Services_CT_Scan
2                            Children
3                Services_Intravenous
4                          Population
5    Initial_admin_Emergency_Admission
Name: Feature, dtype: object
```

In [38]:
```
#Variable Selection
# Checking for the VIF values of the variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor

X = df[['Initial_days', 'Services_CT_Scan', 'Children', 'Services_Intravenous', 'Population', 'Initial_admin_Emergency_Admiss


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

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

print(vif_data)
```

```
                           feature       VIF
0                     Initial_days  1.933906
1                 Services_CT_Scan  1.153794
2                         Children  1.659640
3             Services_Intravenous  1.381602
4                       Population  1.348160
5  Initial_admin_Emergency_Admission  1.667688
```

In [39]:
```
# fit KNN model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score

#Instantiate KNN model
knn = KNeighborsClassifier(n_neighbors=5)

#Fit data to KNN model
knn.fit(X_train, y_train)
print("The accuracy of the KNN model:")
print(knn.score(X_test,y_test))
y_predicted = knn.predict(X_test)
print("The confusion matrix for the KNN model")
print(confusion_matrix(y_test, y_predicted))
y_predicted_probability = knn.predict_proba(X_test)[:,1]
print("The Area Under the Curve (AUC) for the KNN model:")
print(roc_auc_score(y_test, y_predicted_probability))
```

```
The accuracy of the KNN model:
0.8626666666666667
The confusion matrix for the KNN model
[[1684  215]
 [ 197  904]]
The Area Under the Curve (AUC) for the KNN model:
0.9355370841482131
```

In [40]:
```
# Compute classification metrics before scaling
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.90      0.91      1899
           1       0.83      0.85      0.84      1101

    accuracy                           0.88      3000
   macro avg       0.87      0.88      0.88      3000
weighted avg       0.88      0.88      0.88      3000
```

In [41]:
```
#Create KNN model
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier


# Set steps for pipeline object
steps = [('scaler', StandardScaler()),
         ('knn', KNeighborsClassifier())]

# Instantiate pipeline
pipeline = Pipeline(steps)

# Split dataframe
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(X, y, test_size = 0.2, random_state = SEED)

# Scale dataframe with pipeline object
knn_scaled = pipeline.fit(X_train_scaled, y_train_scaled)

# Predict from scaled dataframe
y_pred_scaled = pipeline.predict(X_test_scaled)

# Print new accuracy score of scaled KNN model
print('New accuracy score of scaled KNN model: {:0.3f}'.format(accuracy_score(y_test_scaled, y_pred_scaled)))

# Print new AUC for the scaled KNN model
print("The Area Under the Curve (AUC) for the KNN model:")
print(roc_auc_score(y_test_scaled, y_pred_scaled))
```

```
New accuracy score of scaled KNN model: 0.964
The Area Under the Curve (AUC) for the KNN model:
0.9655695642889258
```

In [42]:
```
#Compute classification metrics after scaling
print(classification_report(y_test_scaled, y_pred_scaled))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1261
           1       0.93      0.97      0.95       739

    accuracy                           0.96      2000
   macro avg       0.96      0.97      0.96      2000
weighted avg       0.96      0.96      0.96      2000
```

Part V: Data Summary and Implications
E1. Explain the accuracy and the area under the curve (AUC) of your classification model.

Classification accuracy is the number of correct predictions made as a ratio of all the predictions (Brownlee, 2018).

The first model had an accuracy score of 86% whereas the scaled model had an accuracy score of 96%. The AUC scores for each model are also very close to the accuracy scores. This shows more evidence that the models are accurate.

The accuracy score shows a ratio of true positives and true negatives, meaning how often we can expect a correct prediction from the model. The aim is to have a score of 1, which is a 100% chance of prediction. Therefore, our scaled model is correct 96% of the time.

E2. Discuss the results and implications of your classification analysis.

Once the data was scaled for k-NN analysis, our accuracy went from 88% to 96%. The implication is that data from new patients could be fed into the model and that re-admissions could be predicted successfully with a 96% rate.

E3. Discuss one limitation of your data analysis.

Choosing the value of k is somewhat arbitrary. I did run GridSearchCV to help choose my k value of 1, however, different k values could produce dramatically different results. GridSearchCV is also heavily computer intensive, this needs to be taken into consideration.

E4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

This model seems to be able to predict with 96% accuracy whether a patient will be re-admitted or not. When new patients are admitted their data should be fed into the model, and the resulting prediction can then be used to categorize the patient and their re-admission risk factor. This information can be useful in coming up with a patient-treament plan. Patients who are predicted for re-admission should be treated with a more intensive care plan than those who are predicted for no re-admission. The intensive care plan can hopefully reduce the chances of future re-admission and save the hospital money.

Part VI: Demonstration
F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=676564fc-c534-4b32-ad09-afbd018423bf

G. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Brownlee, J. (2020, August 15). K-Nearest Neighbors for Machine Learning. Maching Learning Mastery. Retrieved February, 15, 2023 from https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ Bruce, P., Bruce, A., and Gedeck, P. (2020, May 19). Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. O'Reilly.