In [163]:
```
# Standard imports
import numpy as np
import pandas as pd
from pandas import DataFrame

# Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import pyplot as plt
plt.style.use('ggplot')

# Scikit-learn
import sklearn
from sklearn import datasets
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE

 # Import model, splitting method & metrics from sklearn
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
```

In [164]:
```
#load the dataset
df = pd.read_csv("medical clean 1.1.23.csv")
```

In [165]:
```
#examine the first 5 records of data
df.head()
```

Out[165]:

	CaseOrder	Customer_id	Interaction	UID	City	State	
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Mi
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	

5 rows × 50 columns

In [166]:
```
#view describe
df.info
```

Out[166]:
```
<bound method DataFrame.info of       CaseOrder Customer_id
Interaction  \
0             1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1             2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2             3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
3             4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4             5     C544523  5885f56b-d6da-43a3-8760-83583af94266
...         ...         ...                                   ...
9995       9996     B863060  a25b594d-0328-486f-a9b9-0567eb0f9723
9996       9997     P712040  70711574-f7b1-4a17-b15f-48c54564b70f
9997       9998     R778890  1d79569d-8e0f-4180-a207-d67ee4527d26
9998       9999     E344109  f5a68e69-2a60-409b-a92f-ac0847b27db0
9999      10000     I569847  bc482c02-f8c9-4423-99de-3db5e62a18d5

                                   UID          City State        County  \
0     3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan
1     176354c5eef714957d486009feabf195      Marianna    FL       Jackson
2     e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha
3     cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca
4     d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William
...                                ...           ...   ...           ...
9995  39184dc28cc038871912ccc4500049e5       Norlina    NC        Warren
9996  3cd124ccd43147404292e883bf9ec55c        Milmay    NJ      Atlantic
9997  41b770aeee97a5b9e7f69c906a8119d7     Southside    TN    Montgomery
9998  2bb491ef5b1beb1fed758cc6885c167a         Quinn    SD    Pennington
9999  95663a202338000abdf7e09311c2a8a1    Coraopolis    PA     Allegheny

        Zip       Lat        Lng  ...  TotalCharge Additional_charges Item1  \
0     35621  34.34960  -86.72508  ...  3726.702860        17939.403420     3
1     32446  30.84513  -85.22907  ...  4193.190458        17612.998120     3
2     57110  43.54321  -96.63772  ...  2434.234222        17505.192460     2
3     56072  43.89744  -93.51479  ...  2127.830423        12993.437350     3
4     23181  37.59894  -76.88958  ...  2113.073274         3716.525786     2
...     ...       ...        ...  ...          ...                 ...   ...
9995  27563  36.42886  -78.23716  ...  6850.942000         8927.642000     3
9996   8340  39.43609  -74.87302  ...  7741.690000        28507.150000     3
9997  37171  36.36655  -87.29988  ...  8276.481000        15281.210000     3
9998  57775  44.10354 -102.01590  ...  7644.483000         7781.678000     5
9999  15108  40.49998  -80.19959  ...  7887.553000        11643.190000     4

     Item2  Item3  Item4  Item5 Item6 Item7 Item8
0        3      2      2      4     3     3     4
1        4      3      4      4     4     3     3
2        4      4      4      3     4     3     3
3        5      5      3      4     5     5     5
4        1      3      3      5     3     4     3
...    ...    ...    ...    ...   ...   ...   ...
9995     2      2      3      4     3     4     2
9996     3      4      2      5     3     4     4
9997     3      3      4      4     2     3     2
9998     5      3      4      4     3     4     3
9999     3      3      2      3     6     4     3

[10000 rows x 50 columns]>
```

In [167]:
```
#descriptive stats
df.describe()
```

Out[167]:

	CaseOrder	Zip	Lat	Lng	Population	Children	
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.097200	53
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.163659	20
min	1.00000	610.000000	17.967190	-174.209700	0.000000	0.000000	18
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	36
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89

8 rows × 23 columns

In [168]:
```
#check for null values
df.isnull()
```

Out[168]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	To
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	False	False	False	False	False	False	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	...	
9996	False	False	False	False	False	False	False	False	False	False	...	
9997	False	False	False	False	False	False	False	False	False	False	...	
9998	False	False	False	False	False	False	False	False	False	False	...	
9999	False	False	False	False	False	False	False	False	False	False	...	

10000 rows × 50 columns

In [169]:
```
# Getting data types of features
df.dtypes
```

Out[169]:
```
CaseOrder               int64
Customer_id            object
Interaction            object
UID                    object
City                   object
State                  object
County                 object
Zip                     int64
Lat                   float64
Lng                   float64
Population              int64
Area                   object
TimeZone               object
Job                    object
Children                int64
Age                     int64
Income                float64
Marital                object
Gender                 object
ReAdmis                object
VitD_levels           float64
Doc_visits              int64
Full_meals_eaten        int64
vitD_supp               int64
Soft_drink             object
Initial_admin          object
HighBlood              object
Stroke                 object
Complication_risk      object
Overweight             object
Arthritis              object
Diabetes               object
Hyperlipidemia         object
BackPain               object
Anxiety                object
Allergic_rhinitis      object
Reflux_esophagitis     object
Asthma                 object
Services               object
Initial_days          float64
TotalCharge           float64
Additional_charges    float64
Item1                   int64
Item2                   int64
Item3                   int64
Item4                   int64
Item5                   int64
Item6                   int64
Item7                   int64
Item8                   int64
dtype: object
```

In [170]:
```
#change to integers
df['TotalCharge'] = df['TotalCharge'].astype(int)
df['Initial_days'] = df['Initial_days'].astype(int)
#Change object to category
df["Gender"] = df["Gender"].astype('category')
df["ReAdmis"] = df["ReAdmis"].astype('category')
df["Soft_drink"] = df["Soft_drink"].astype('category')
df["Initial_admin"] = df["Initial_admin"].astype('category')
df["HighBlood"] = df["HighBlood"].astype('category')
df["Stroke"] = df["Stroke"].astype('category')
df["Overweight"] = df["Overweight"].astype('category')
df["Arthritis"] = df["Arthritis"].astype('category')
df["Diabetes"] = df["Diabetes"].astype('category')
df["Hyperlipidemia"] = df["Hyperlipidemia"].astype('category')
df["BackPain"] = df["BackPain"].astype('category')
df["Anxiety"] = df["Anxiety"].astype('category')
df["Allergic_rhinitis"] = df["Allergic_rhinitis"].astype('category')
df["Reflux_esophagitis"] = df["Reflux_esophagitis"].astype('category')
df["Services"] = df["Services"].astype('category')
df["Asthma"] = df["Asthma"].astype('category')
df["Marital"] = df["Marital"].astype('category')
df["Complication_risk"] = df["Complication_risk"].astype('category')
```

In [171]:
```
#drop columns not being used
to_drop = ['CaseOrder', 'Customer_id', 'Marital', 'Age', 'Hyperlipidemia', 'Asthm
df.drop(to_drop, inplace=True, axis=1)
```

In [172]:
```
#check data types
df.dtypes
```

Out[172]:
```
ReAdmis             category
Doc_visits             int64
Full_meals_eaten       int64
Soft_drink          category
HighBlood           category
Stroke              category
Overweight          category
Arthritis           category
Diabetes            category
BackPain            category
Anxiety             category
dtype: object
```

In [173]:
```
dummies on categorical
mmies(df, columns = ['ReAdmis', 'HighBlood', 'Overweight','Soft_drink', 'Stroke',
```

Out[173]:

	Doc_visits	Full_meals_eaten	ReAdmis_No	ReAdmis_Yes	HighBlood_No	HighBlood_Yes	Ov
0	6	0	1	0	0	1	
1	4	2	1	0	0	1	
2	4	1	1	0	0	1	
3	4	1	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	
9995	4	2	1	0	0	1	
9996	5	0	0	1	0	1	
9997	4	2	0	1	0	1	
9998	5	2	0	1	1	0	
9999	5	0	0	1	1	0	

10000 rows × 20 columns

In [174]:
```
the get dummies responses
mmies(df, columns = ['ReAdmis', 'HighBlood', 'Overweight','Soft_drink', 'Stroke',
```

In [175]:
```
#check data types
df_ready.dtypes
```

Out[175]:
```
Doc_visits          int64
Full_meals_eaten    int64
ReAdmis_No          uint8
ReAdmis_Yes         uint8
HighBlood_No        uint8
HighBlood_Yes       uint8
Overweight_No       uint8
Overweight_Yes      uint8
Soft_drink_No       uint8
Soft_drink_Yes      uint8
Stroke_No           uint8
Stroke_Yes          uint8
Arthritis_No        uint8
Arthritis_Yes       uint8
Diabetes_No         uint8
Diabetes_Yes        uint8
BackPain_No         uint8
BackPain_Yes        uint8
Anxiety_No          uint8
Anxiety_Yes         uint8
dtype: object
```

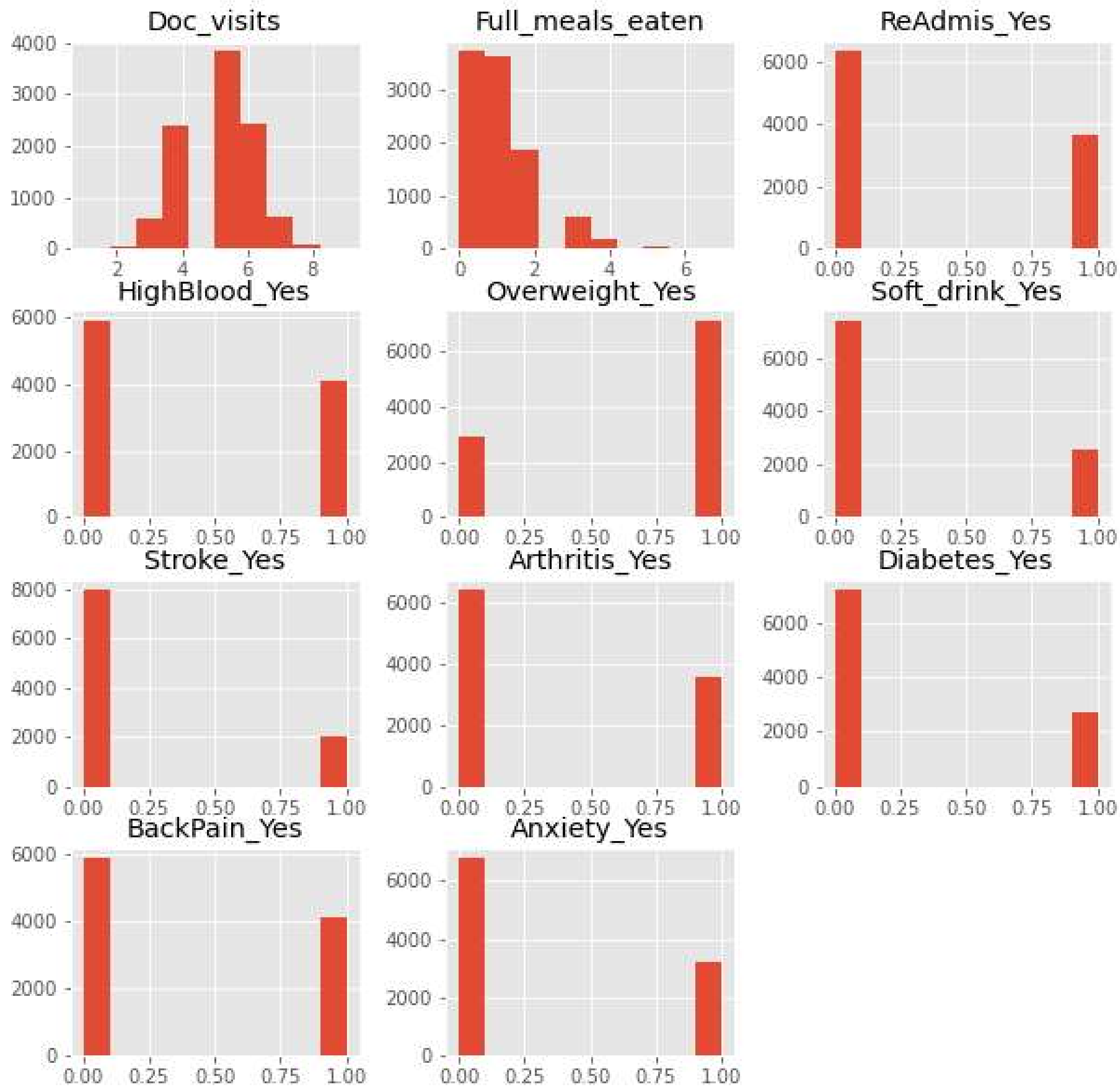
In [176]:
```
#drop multiple columns by name
df_ready.drop(['ReAdmis_No', 'HighBlood_No', 'Overweight_No', 'Soft_drink_No', 'A
```

In [177]:
```
#check data types
df_ready.dtypes
```

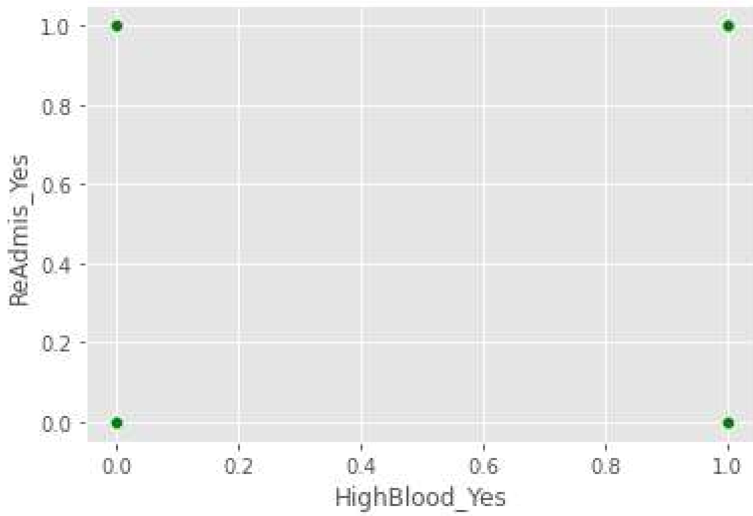
Out[177]:
```
Doc_visits          int64
Full_meals_eaten    int64
ReAdmis_Yes         uint8
HighBlood_Yes       uint8
Overweight_Yes      uint8
Soft_drink_Yes      uint8
Stroke_Yes          uint8
Arthritis_Yes       uint8
Diabetes_Yes        uint8
BackPain_Yes        uint8
Anxiety_Yes         uint8
dtype: object
```

In [178]:
```
#view histograms to get a feel for the data
plt.style.use('ggplot')

X = df_ready.drop('ReAdmis_Yes', 1).values

# drop target variable
y1 = df_ready['ReAdmis_Yes'].values
pd.DataFrame.hist(df_ready, figsize = [10,10]);
plt.show()
```

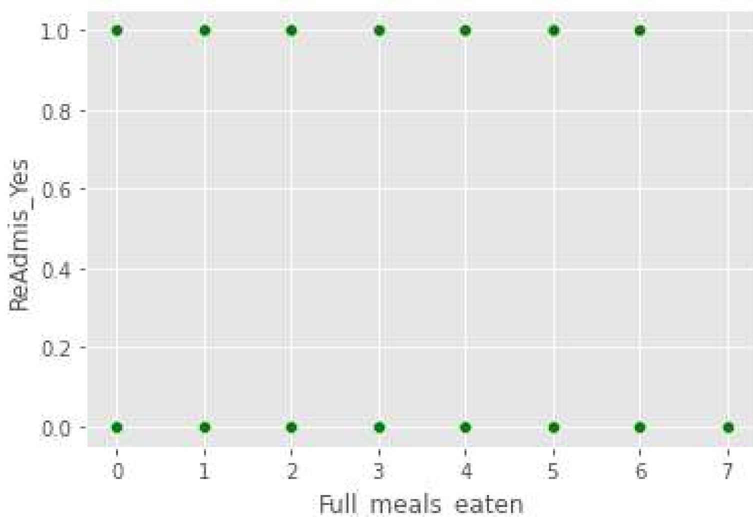
```
C:\Users\Brittany\AppData\Local\Temp\ipykernel_14284\3692946228.py:4: FutureWar
ning: In a future version of pandas all arguments of DataFrame.drop except for
the argument 'labels' will be keyword-only.
  X = df_ready.drop('ReAdmis_Yes', 1).values
```



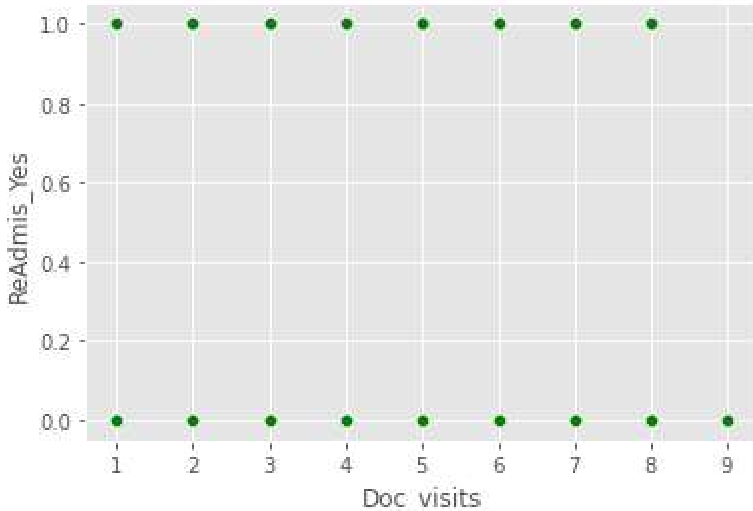
In [179]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['HighBlood_Yes'], y=df_ready['ReAdmis_Yes'], color='gr
plt.show()
```



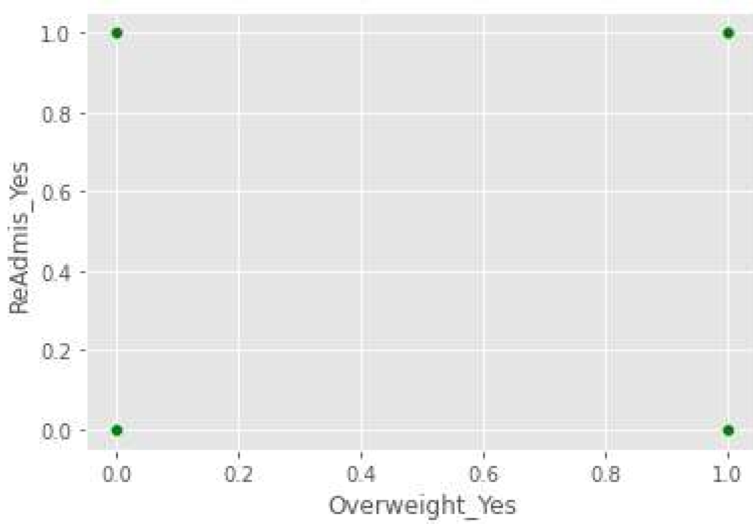
In [180]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Full_meals_eaten'], y=df_ready['ReAdmis_Yes'], color=
plt.show()
```



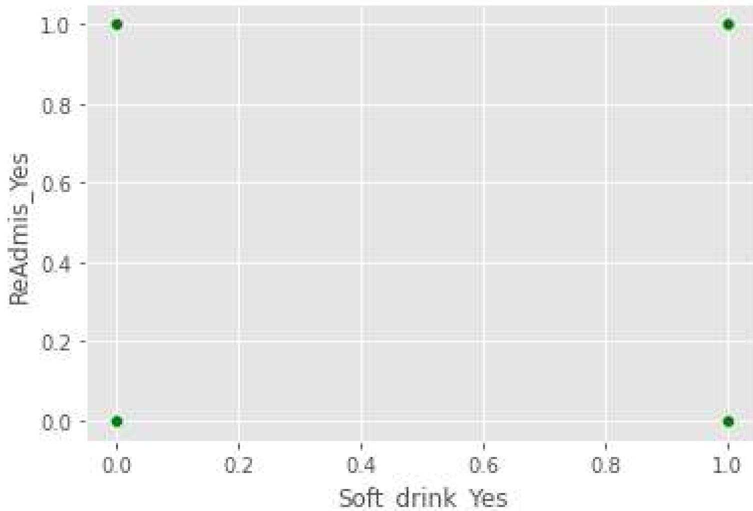
In [181]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Doc_visits'], y=df_ready['ReAdmis_Yes'], color='greer
plt.show()
```



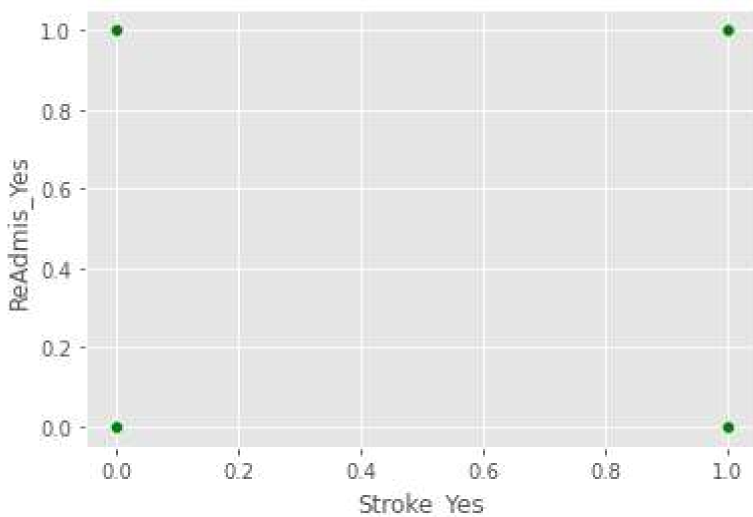
In [182]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Overweight_Yes'], y=df_ready['ReAdmis_Yes'], color='g
plt.show()
```



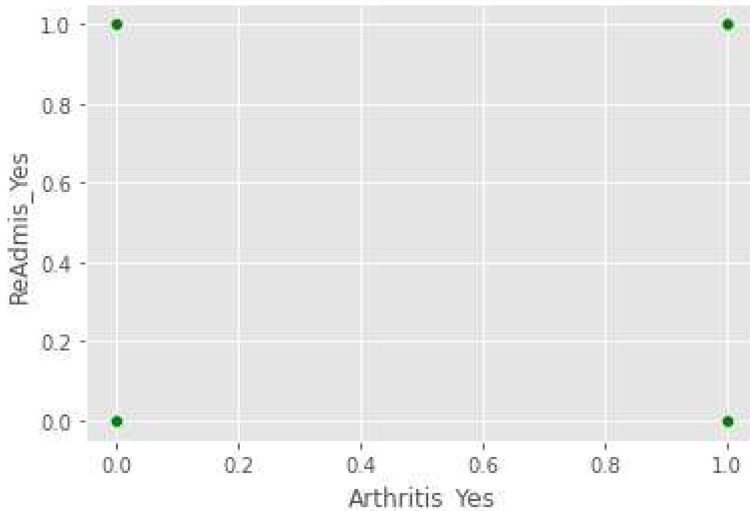
In [183]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Soft_drink_Yes'], y=df_ready['ReAdmis_Yes'], color='g
plt.show()
```



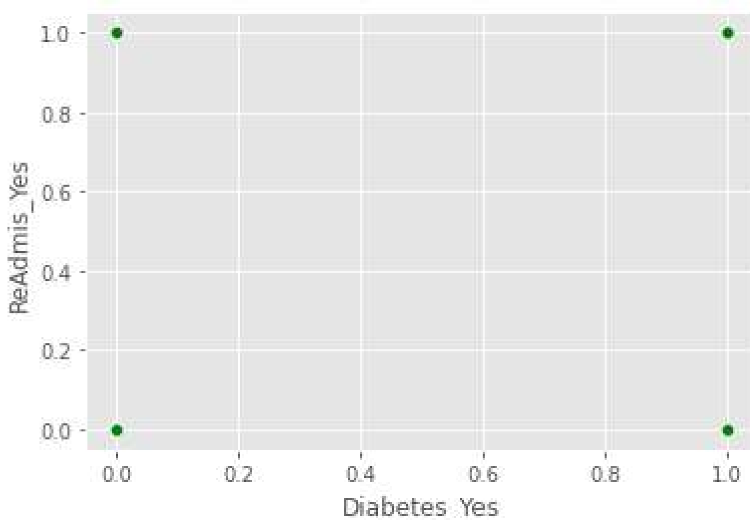
In [184]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Stroke_Yes'], y=df_ready['ReAdmis_Yes'], color='green
plt.show()
```



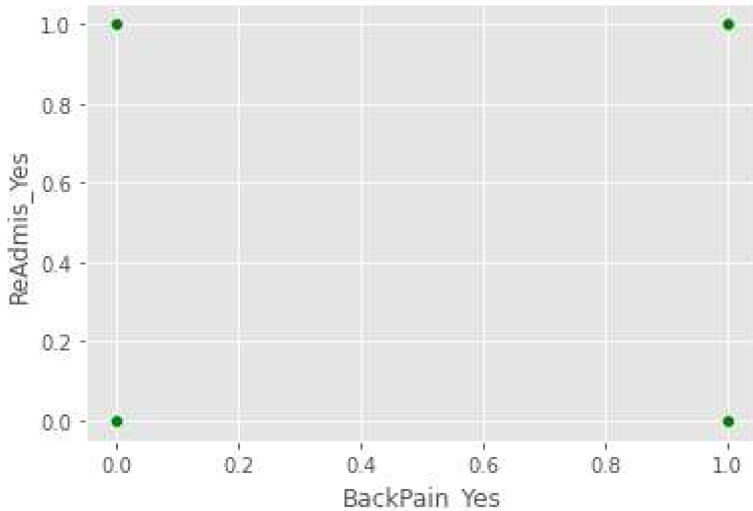
In [185]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Arthritis_Yes'], y=df_ready['ReAdmis_Yes'], color='gr
plt.show()
```



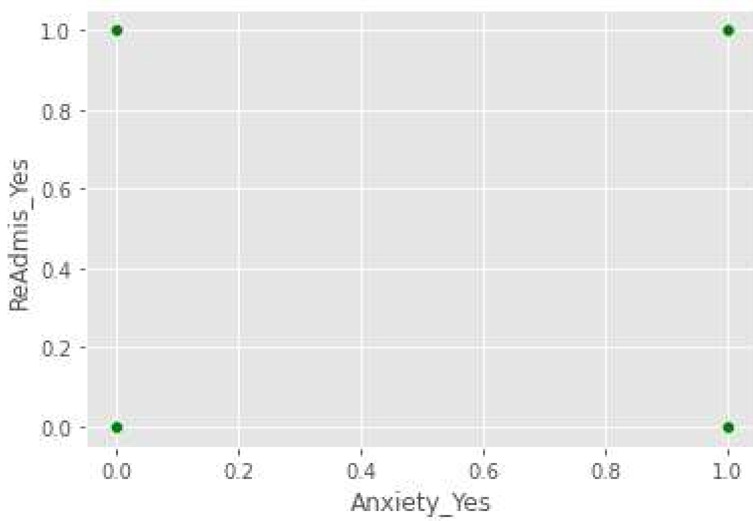
In [186]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Diabetes_Yes'], y=df_ready['ReAdmis_Yes'], color='gre
plt.show()
```



In [187]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['BackPain_Yes'], y=df_ready['ReAdmis_Yes'], color='gre
plt.show()
```

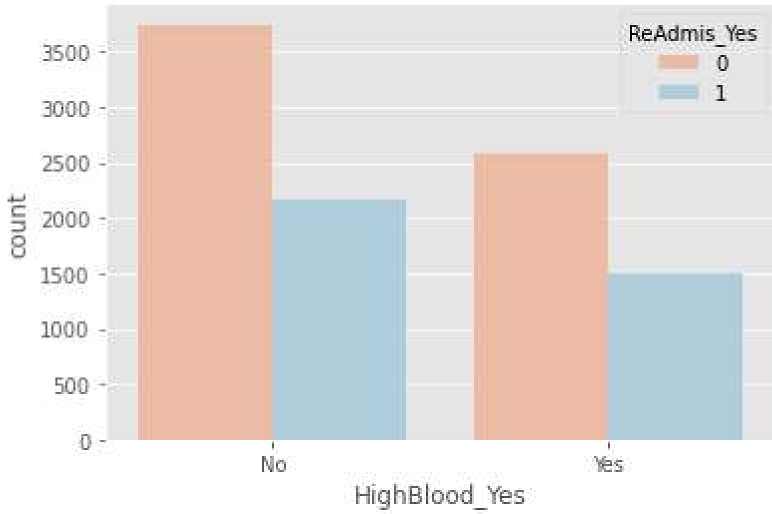


In [188]:
```
# A scatterplot to get an idea of correlations between potentially related variab
sns.scatterplot(x=df_ready['Anxiety_Yes'], y=df_ready['ReAdmis_Yes'], color='gree
plt.show()
```

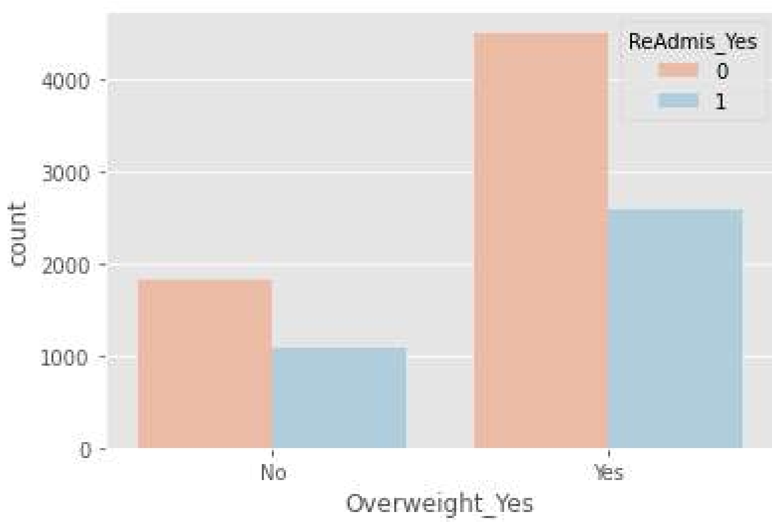


In [189]:
```
# set the plot style to ggplot
plt.style.use('ggplot')
```

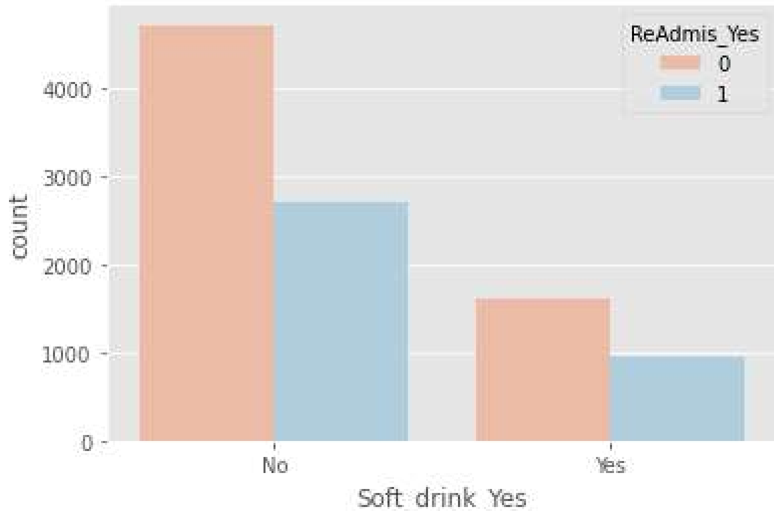
In [190]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='HighBlood_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu'
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



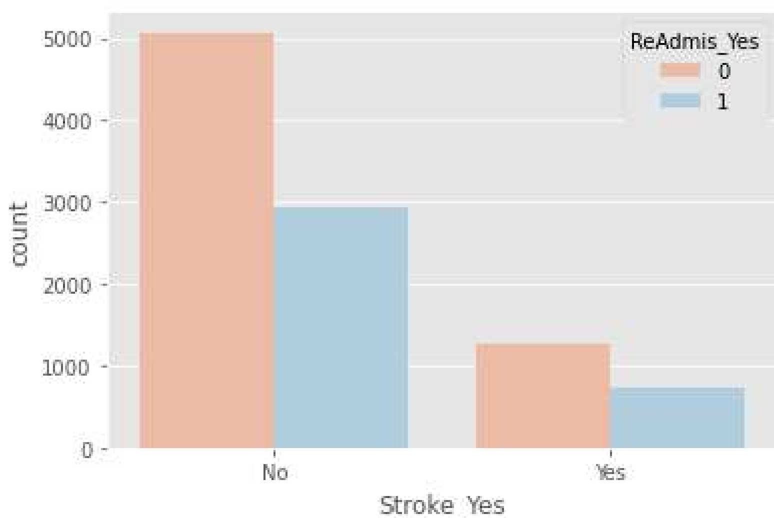
In [191]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='Overweight_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



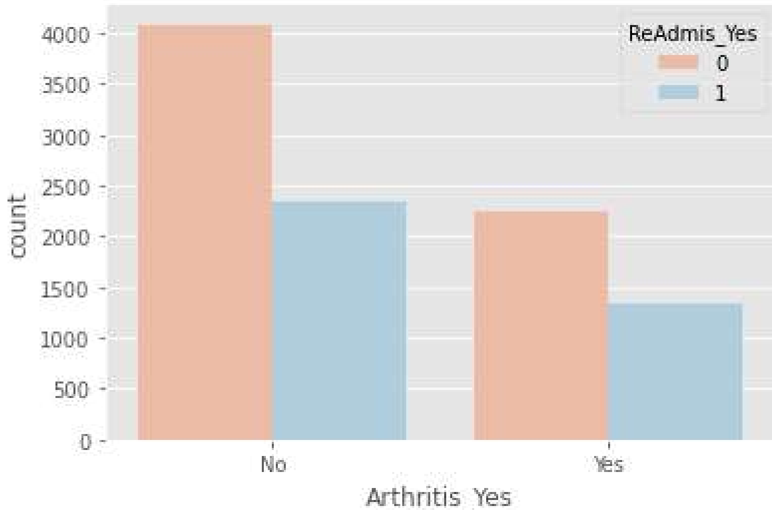
In [192]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='Soft_drink_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



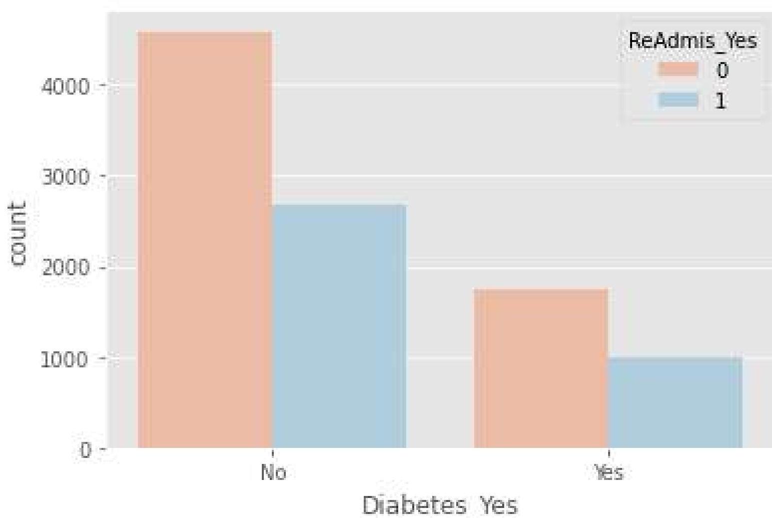
In [193]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='Stroke_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



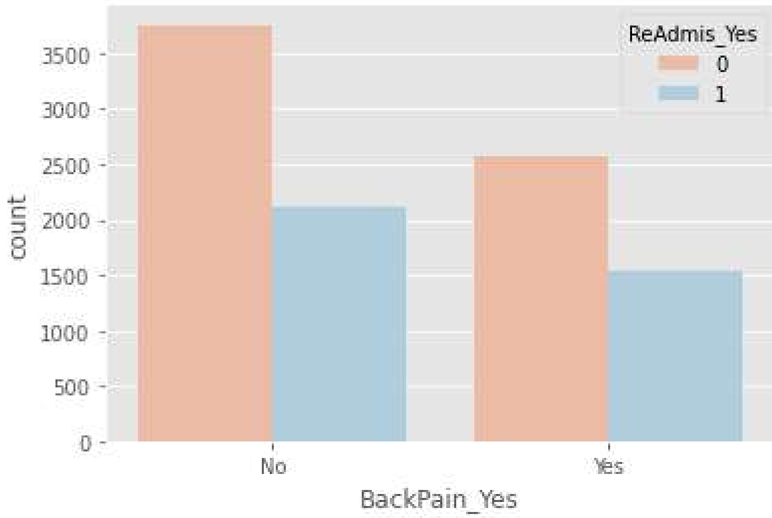
In [194]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='Arthritis_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu'
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



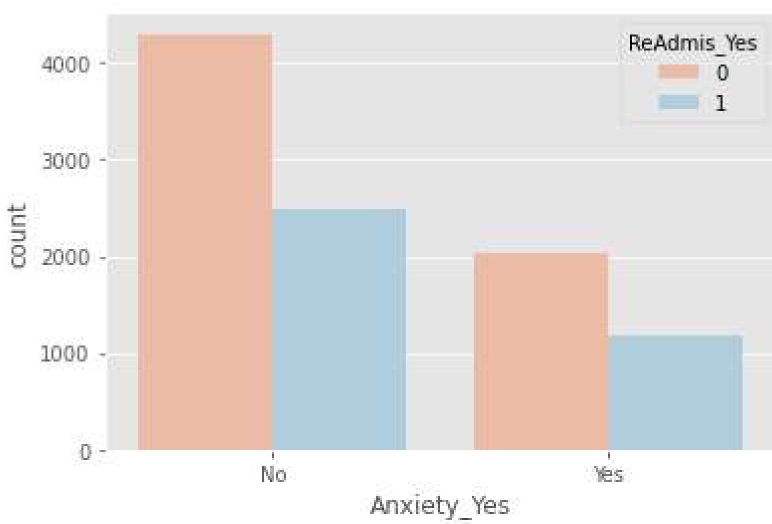
In [195]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='Diabetes_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



In [196]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='BackPain_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```

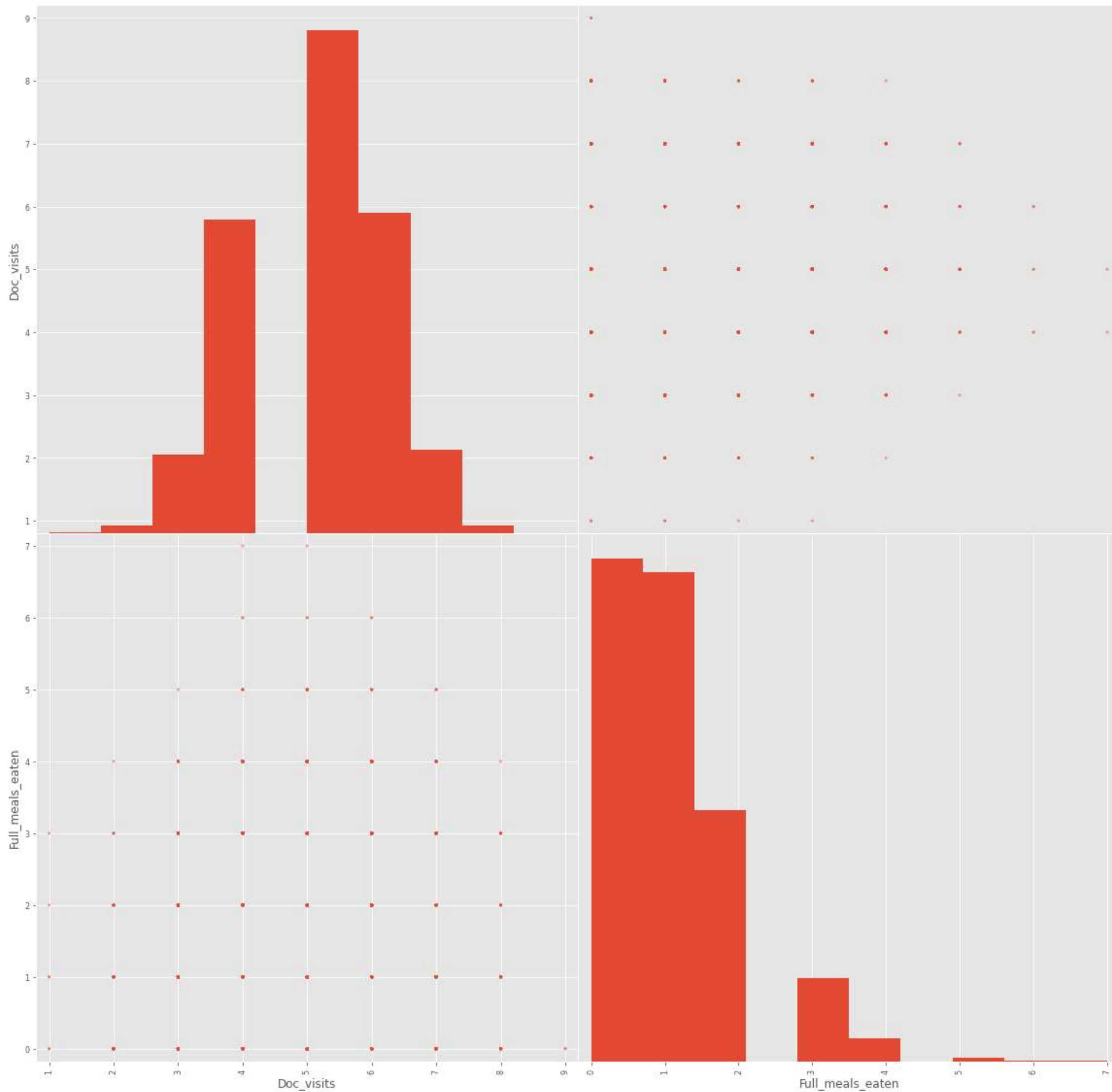


In [197]:
```
# Countplots of categorical variables
plt.figure()
sns.countplot(x='Anxiety_Yes', hue='ReAdmis_Yes', data=df_ready, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



In [198]:
```
# A scatter matrix of the discrete variables for high level overview of potential
df_discrete = df_ready[['Doc_visits', 'Full_meals_eaten']]
pd.plotting.scatter_matrix(df_discrete, figsize = [20, 20])
```

Out[198]:
```
array([[<AxesSubplot:xlabel='Doc_visits', ylabel='Doc_visits'>,
        <AxesSubplot:xlabel='Full_meals_eaten', ylabel='Doc_visits'>],
       [<AxesSubplot:xlabel='Doc_visits', ylabel='Full_meals_eaten'>,
        <AxesSubplot:xlabel='Full_meals_eaten', ylabel='Full_meals_eaten'>]],
      dtype=object)
```



In [199]:
```
#save prepared data
df_ready.to_csv('Documents/PreparedData D209 Task2.csv')
```

In [200]:
```
# List features for analysis
features = (list(df_ready.columns[:-1]))
print('Features for analysis include: \n', features)
```

```
Features for analysis include: 
 ['Doc_visits', 'Full_meals_eaten', 'ReAdmis_Yes', 'HighBlood_Yes', 'Overweight
_Yes', 'Soft_drink_Yes', 'Stroke_Yes', 'Arthritis_Yes', 'Diabetes_Yes', 'BackPa
in_Yes']
```

In [201]:
```
# Re-read fully numerical prepared dataset
df_ready = pd.read_csv("PreparedData D209 Task2.csv")
```

In [202]:
```
# Set predictor features & target variable
X = df_ready.drop('ReAdmis_Yes', axis=1).values
y = df_ready['ReAdmis_Yes'].values
```

In [203]:
```
# Create training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, rando
```

In [204]:
```
# Export y_test dataset
y_test_df_ready = pd.DataFrame(X_test)
y_test_df_ready.to_csv('Documents/PreparedData D209 y_test Task2.csv')
```

In [205]:
```
# Export y train  dataset
y_train_df_ready = pd.DataFrame(X_train)
y_train_df_ready.to_csv('Documents/PreparedData D209 y_train Task2.csv')
```

In [206]:
```
# Export X train  dataset
X_train_df_ready = pd.DataFrame(X_train)
X_train_df_ready.to_csv('Documents/PreparedData D209 X_train Task2.csv')
```

In [207]:
```
# Export X test  dataset
X_test_df_ready = pd.DataFrame(X_test)
X_test_df_ready.to_csv('Documents/PreparedData D209 X_test Task2.csv')
```

In [267]:
```
# Instantiate Decision Tree Regressor model
dt = DecisionTreeRegressor(max_depth = 8, min_samples_leaf = 0.1, random_state =
```

In [268]:
```
# Fit dataframe to Decision Tree Regressor model
dt.fit(X_train, y_train)
```

Out[268]:
```
DecisionTreeRegressor

DecisionTreeRegressor(max_depth=8, min_samples_leaf=0.1, random_state=1)
```

In [269]:
```
# Predict Outcomes from test set
y_pred = dt.predict(X_test)
```

In [270]:
```
# Compute test set MSE
mse_dt = MSE(y_test, y_pred)
```

In [271]:
```
# Compute test set RMSE
rmse_dt = mse_dt**(1/2)
```

In [272]:
```
# Print initial RMSE
print('Initial RMSE score Decision Tree Regressor model: {:.3f}'.format(rmse_dt))
```

```
Initial RMSE score Decision Tree Regressor model: 0.302
```

In [273]:
```
# Compute the coefficient of determination (R-squared)
scores = cross_val_score(dt, X, y, scoring='r2')
```

In [274]:
```
# Print R-squared value
print('Cross validation R-squared values: ', scores)
```

```
Cross validation R-squared values:  [ 1.          1.          0.56957248 -0.005
45571 -0.00253707]
```

In [275]:
```
# Print Mean Squared Error
print('With a manual calculation, the Mean Squared Error: {:.3f} '.format(sum(abs
```

```
With a manual calculation, the Mean Squared Error: 0.091
```

In [276]:
```
print('Using scikit-lean, the Mean Squared Error: {:.3f}'.format(MSE(y_test, y_pr
```

```
Using scikit-lean, the Mean Squared Error: 0.091
```

In [277]:
```
# Calculate & print the Root Mean Squared Error
RMSE = MSE(y_test, y_pred)**(1/2)
```

In [278]:
```
# Print the Root Mean Squared Error
print('Root Mean Squared Error: {:.3f} '.format(RMSE))
```

```
Root Mean Squared Error: 0.302
```

In [279]:
```
# Get parameters of Decision Tree Regression model for cross validation
dt.get_params()
```

Out[279]:
```
{'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': 8,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 0.1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 1,
 'splitter': 'best'}
```

In [280]:
```
# Define grid of hyperparameters
params_dt = {'max_depth': [4, 6, 8],
 'min_samples_leaf': [0.1, 0.2],
 'max_features': ['log2', 'sqrt']}
```

In [281]:
```
# Re-instantiate Decision Tree Regressor for cross validation
dt = DecisionTreeRegressor()
```

In [282]:
```
# Instantiate GridSearch cross validation
dt_cv = GridSearchCV(estimator=dt,
 param_grid=params_dt,
 scoring='neg_mean_squared_error',
 cv=5,
 verbose=1,
 n_jobs=-1)
```

In [283]:
```
# Fit model to
dt_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

Out[283]:

```
▸ GridSearchCV
▸ estimator: DecisionTreeRegressor
    ▸ DecisionTreeRegressor
```

In [284]:
```
# Print best parameters
print('Best parameters for this Decision Tree Regressor model: {}'.format(dt_cv.b
```

```
Best parameters for this Decision Tree Regressor model: {'max_depth': 4, 'max_f
eatures': 'log2', 'min_samples_leaf': 0.2}
```

In [285]:
```
print('Best score for this Decision Tree Regressor model: {:.3f}'.format(dt_cv.be
```

```
Best score for this Decision Tree Regressor model: -0.122
```

In [ ]: