Eric Yarger, Predictive Analysis

In [1]:
```
# Import Libraries

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import missingno as msno
from scipy import stats
from scipy.stats import zscore
```

In [2]:
```
# Jupyter environment version
!jupyter --version
```

```
jupyter core     : 4.6.3
jupyter-notebook : 6.0.3
qtconsole        : 4.7.2
ipython          : 7.13.0
ipykernel        : 5.1.4
jupyter client   : 6.1.2
jupyter lab      : 1.2.6
nbconvert        : 5.6.1
ipywidgets       : 7.5.1
nbformat         : 5.0.4
traitlets        : 4.3.3
```

In [3]:
```
# Python Environment version
import platform
print(platform.python_version())
```

```
3.7.7
```

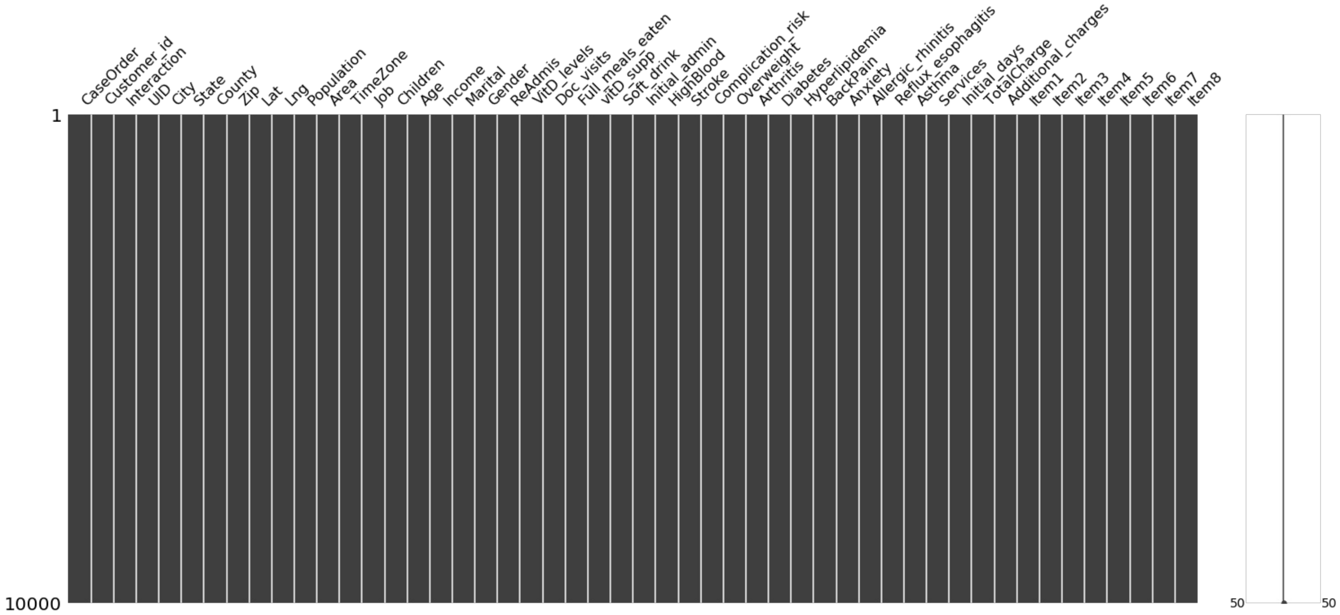
Data Preparation

Step 1 : Load the Data and initial visualization

In [4]:
```
df = pd.read_csv('C:/Users/ericy/Desktop/medical_clean.csv')
```

In [5]:
```
msno.matrix(df)
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1925b98e888>
```



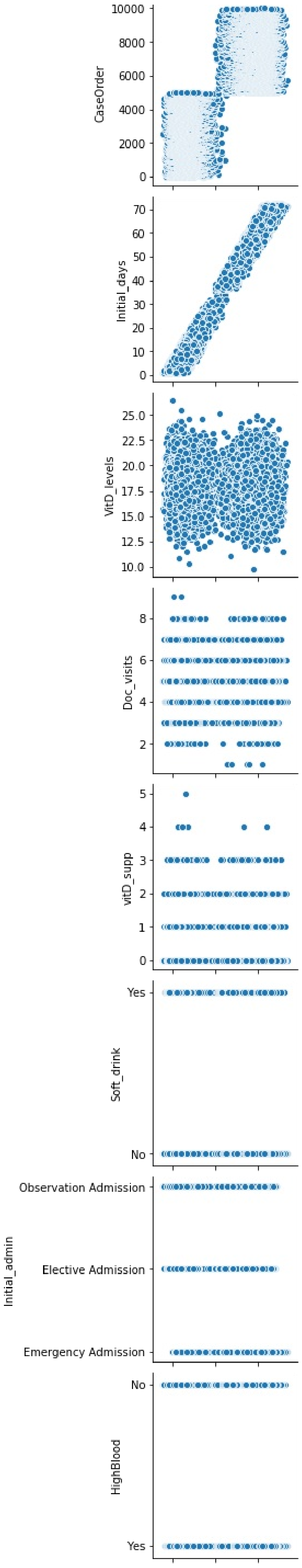
In [ ]:

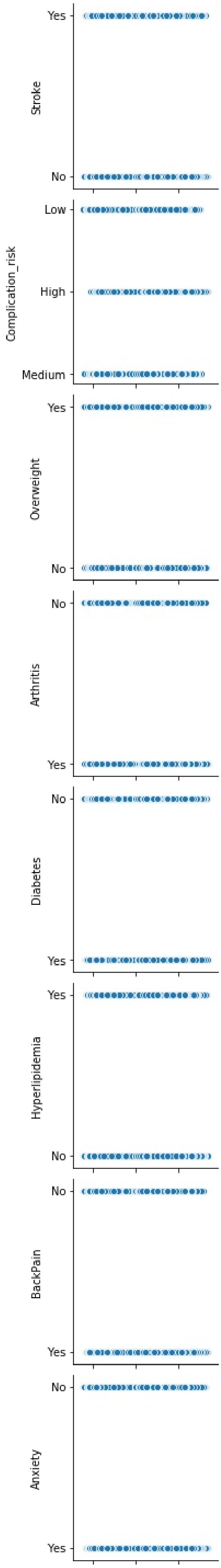
Step 2 : Rename columns and create Pairplots

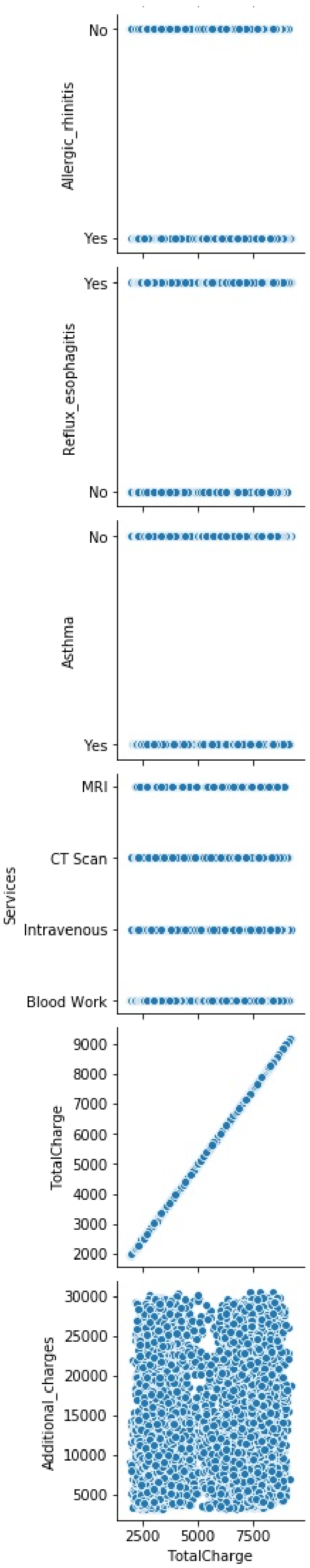
In [6]:
```
df.rename(columns={'Item1':'Timely_admis','Item2':'Timely_treat','Item3':'Timely_vis','Item4':'Reliability','It
```

In [7]:
```
sns.pairplot(df, x_vars=['TotalCharge'], y_vars=['CaseOrder','Initial_days','VitD_levels','Doc_visits','vitD_su
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1925cef2f88>
```

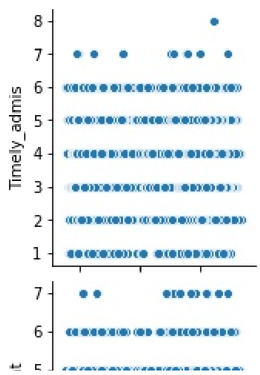

CaseOrder
Initial_days
VitD_levels
Doc_visits
vitD_supp
Soft_drink
Yes
No
Initial_admin
Observation Admission
Elective Admission
Emergency Admission
HighBlood
No
Yes

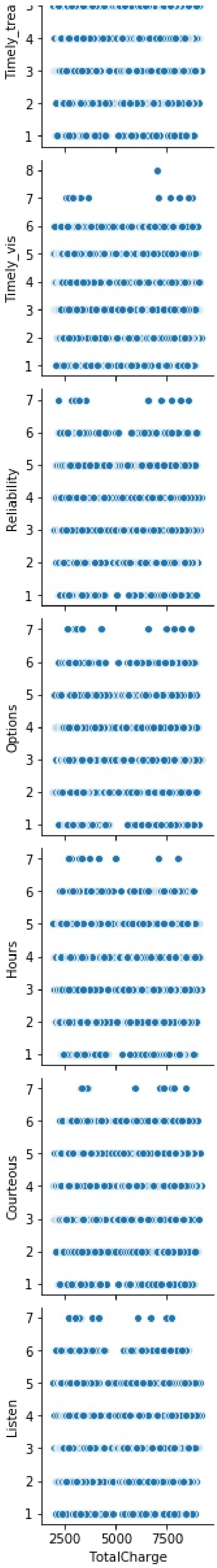

Stroke
Yes
No
Complication_risk
Low
High
Medium
Overweight
Yes
No
Arthritis
No
Yes
Diabetes
No
Yes
Hyperlipidemia
Yes
No
BackPain
No
Yes
Anxiety
No
Yes



In [8]:
```
sns.pairplot(df, x_vars=['TotalCharge'], y_vars=['Timely_admis','Timely_treat','Timely_vis','Reliability','Opti
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x1925d920748>





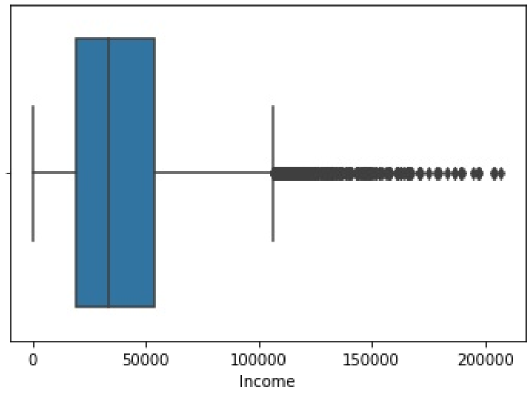
Step 3 : Address missing data, duplicates, and outliers.

In [9]:
```
# Calculate Z-scores, remove Outliers Z > 3

#df.isnull().sum()
```

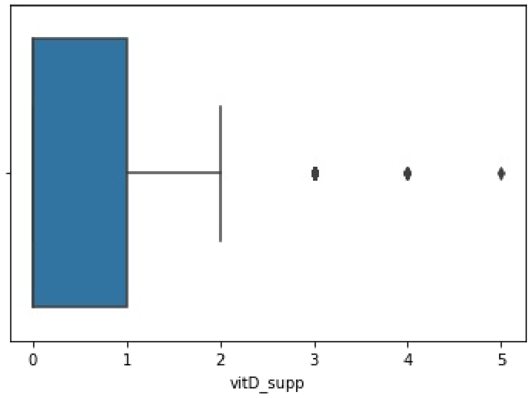
In [10]:
```
df.duplicated().any()
```

Out[10]:
```
False
```

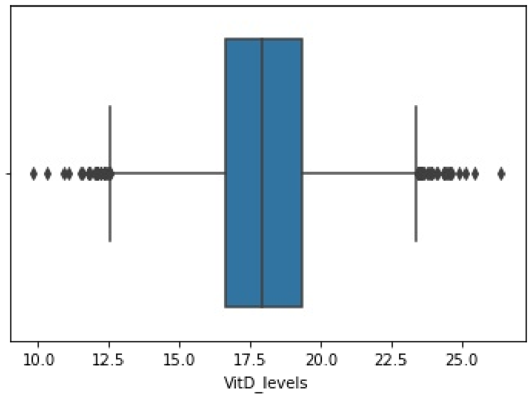
In [11]:
```
sns.boxplot(df['Income'])
```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f076948>



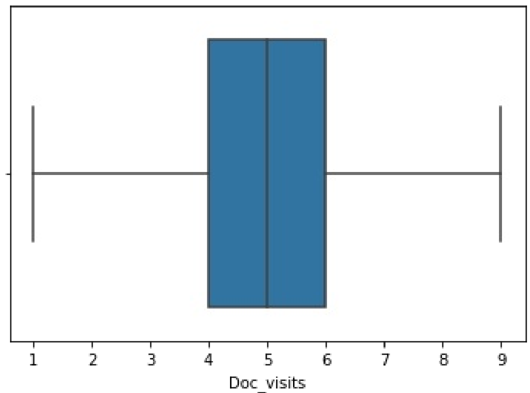
In [12]:
```
sns.boxplot(df['vitD_supp'])
```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f0dc248>



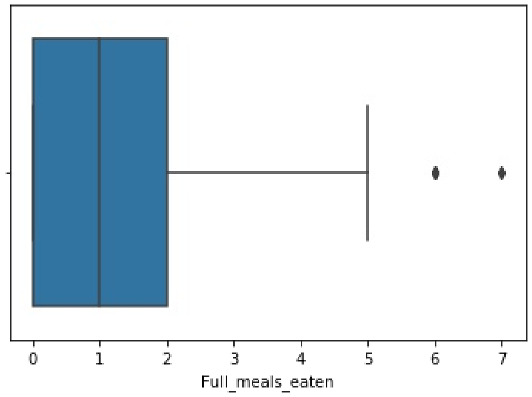
In [13]:
```
sns.boxplot(df['VitD_levels'])
```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f142688>



In [14]:
```
sns.boxplot(df['Doc_visits'])
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f1abb88>

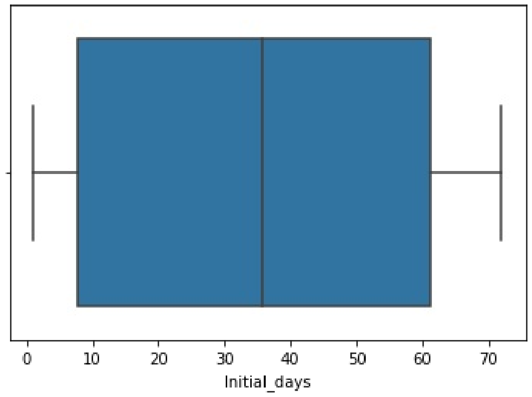


In [15]:
```
sns.boxplot(df['Full_meals_eaten'])
```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f217648>



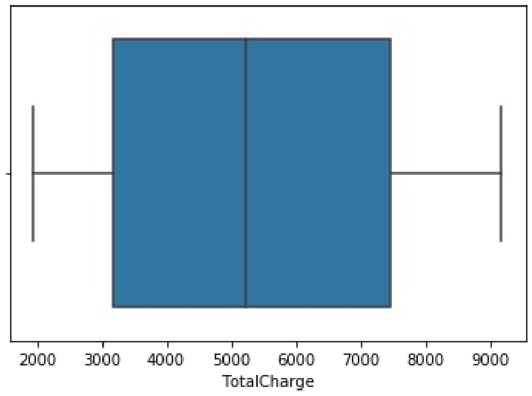
In [16]:
```
sns.boxplot(df['Initial_days'])
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f289c88>



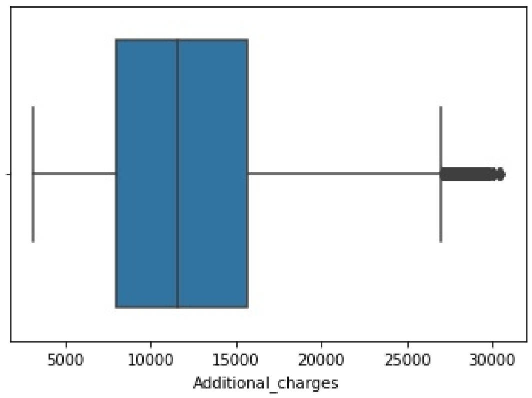
In [17]:
```
sns.boxplot(df['TotalCharge'])
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f2d8808>



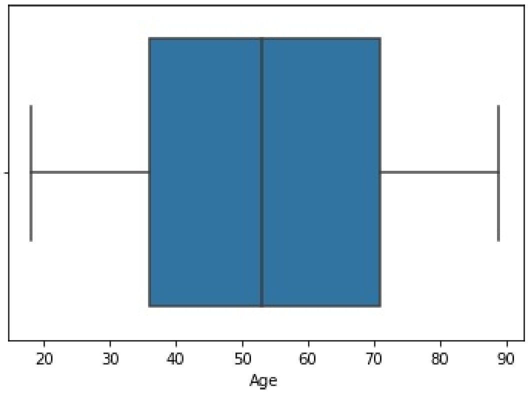
In [18]:
```
sns.boxplot(df['Additional_charges'])
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f344208>



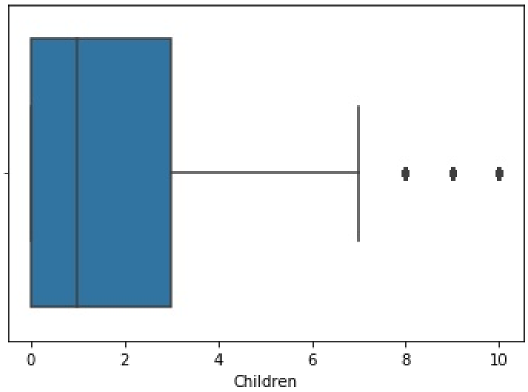
In [19]:
```
sns.boxplot(df['Age'])
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f3ad5c8>



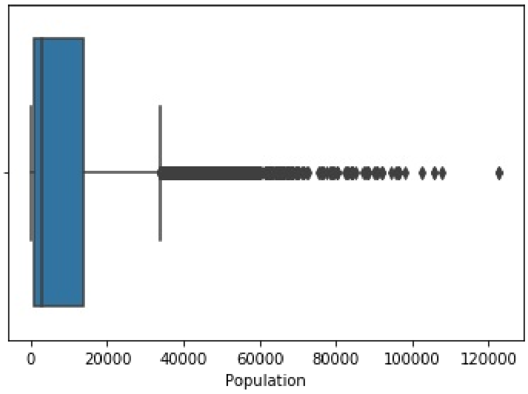
In [20]:
```
sns.boxplot(df['Children'])
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1925f414048>
```



In [21]:
```
sns.boxplot(df['Population'])
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1925f479d08>
```



In [ ]:

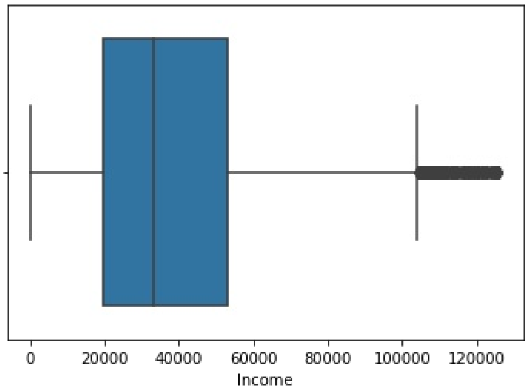
In [22]:
```
# Outlier removal method via Z-score, Code reference (Bushmanov, 2019)

num_data = df.select_dtypes(include=['number'])
cat_data = df.select_dtypes(exclude=['number'])
```

In [23]:
```
idx = np.all(stats.zscore(num_data) <3, axis=1)
```

In [24]:
```
df = pd.concat([num_data.loc[idx], cat_data.loc[idx]], axis=1)
```

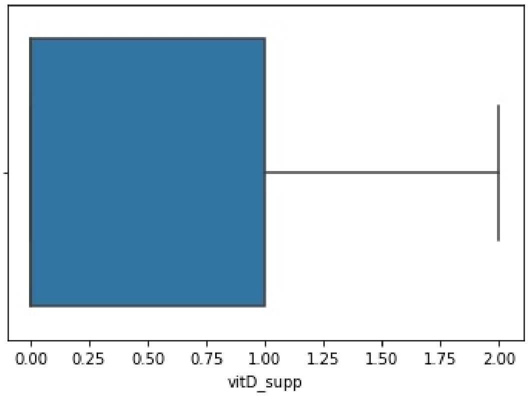
In [25]:
```
sns.boxplot(df['Income'])
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1925f4fcc48>
```



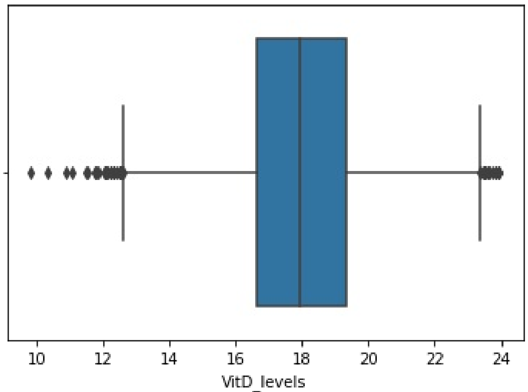
In [26]:
```
sns.boxplot(df['vitD_supp'])
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f8aca08>



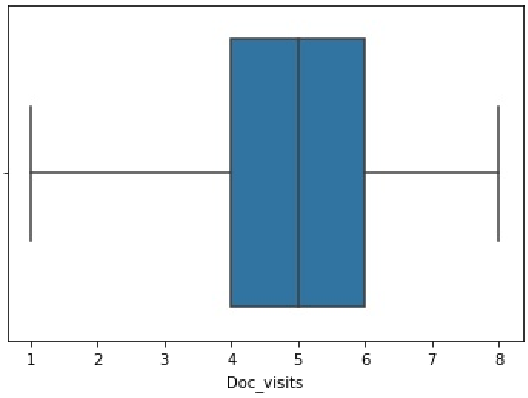
In [27]:
```
sns.boxplot(df['VitD_levels'])
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f920c88>

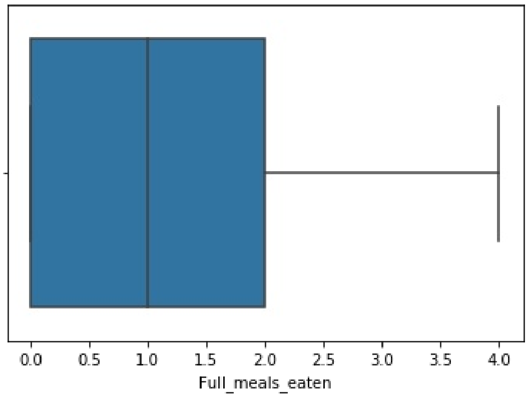


In [28]:
```
sns.boxplot(df['Doc_visits'])
```

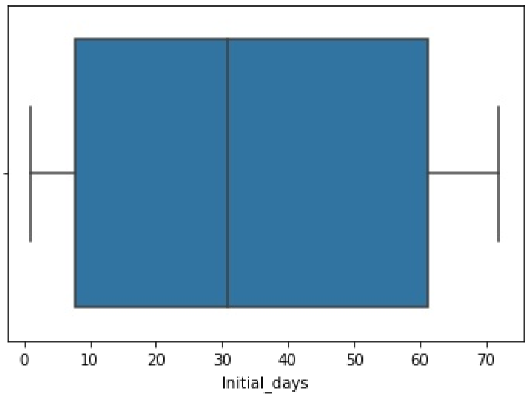
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f9acdc8>



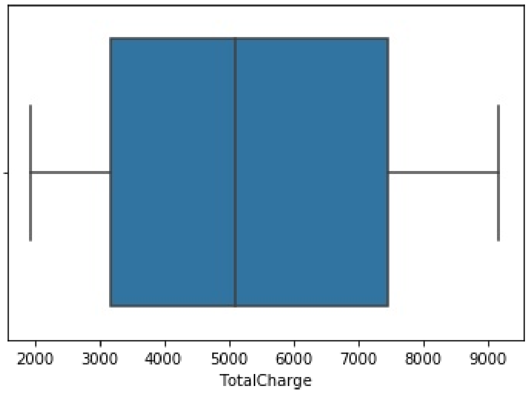
In [29]:
```
sns.boxplot(df['Full_meals_eaten'])
```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1925f9f6f48>



In [30]:
```
sns.boxplot(df['Initial_days'])
```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1925fa71088>

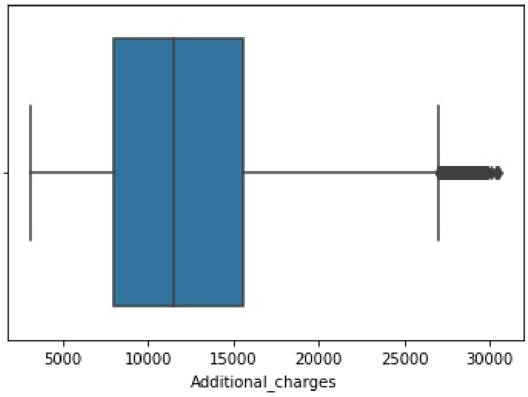


In [31]:
```
sns.boxplot(df['TotalCharge'])
```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1925fae44c8>



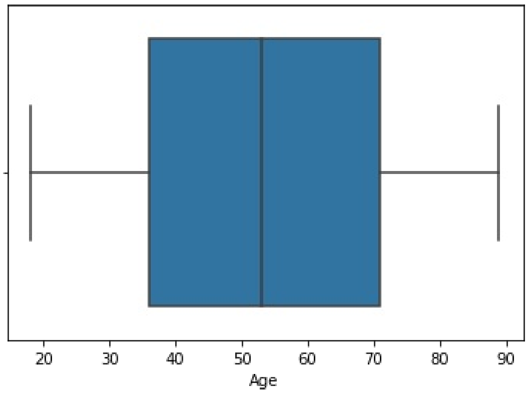
In [32]:
```
sns.boxplot(df['Additional_charges'])
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1925fb4b488>



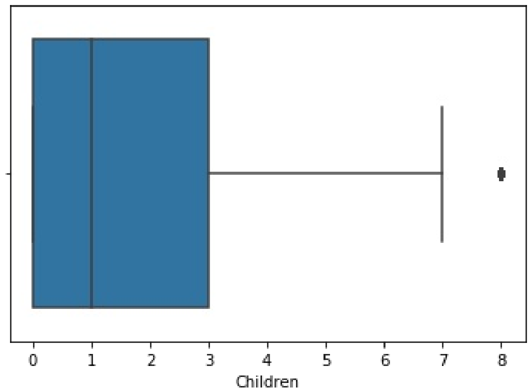
In [33]:
```
sns.boxplot(df['Age'])
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x19260da1708>



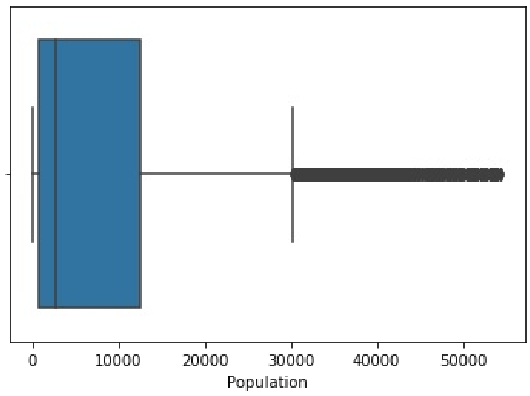
In [34]:
```
sns.boxplot(df['Children'])
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x19260e0c408>



In [35]:
```
sns.boxplot(df['Population'])
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19260e7b288>
```



In [36]:
```
# replace yes/no with 1/0 ReAdmis
df.replace(('Yes','No'), (1,0), inplace=True)
```

In [37]:
```
df.duplicated().sum()
```

Out[37]:
```
0
```

Step 4 : Look at correlation between variables

In [38]:
```
df.corr()
```

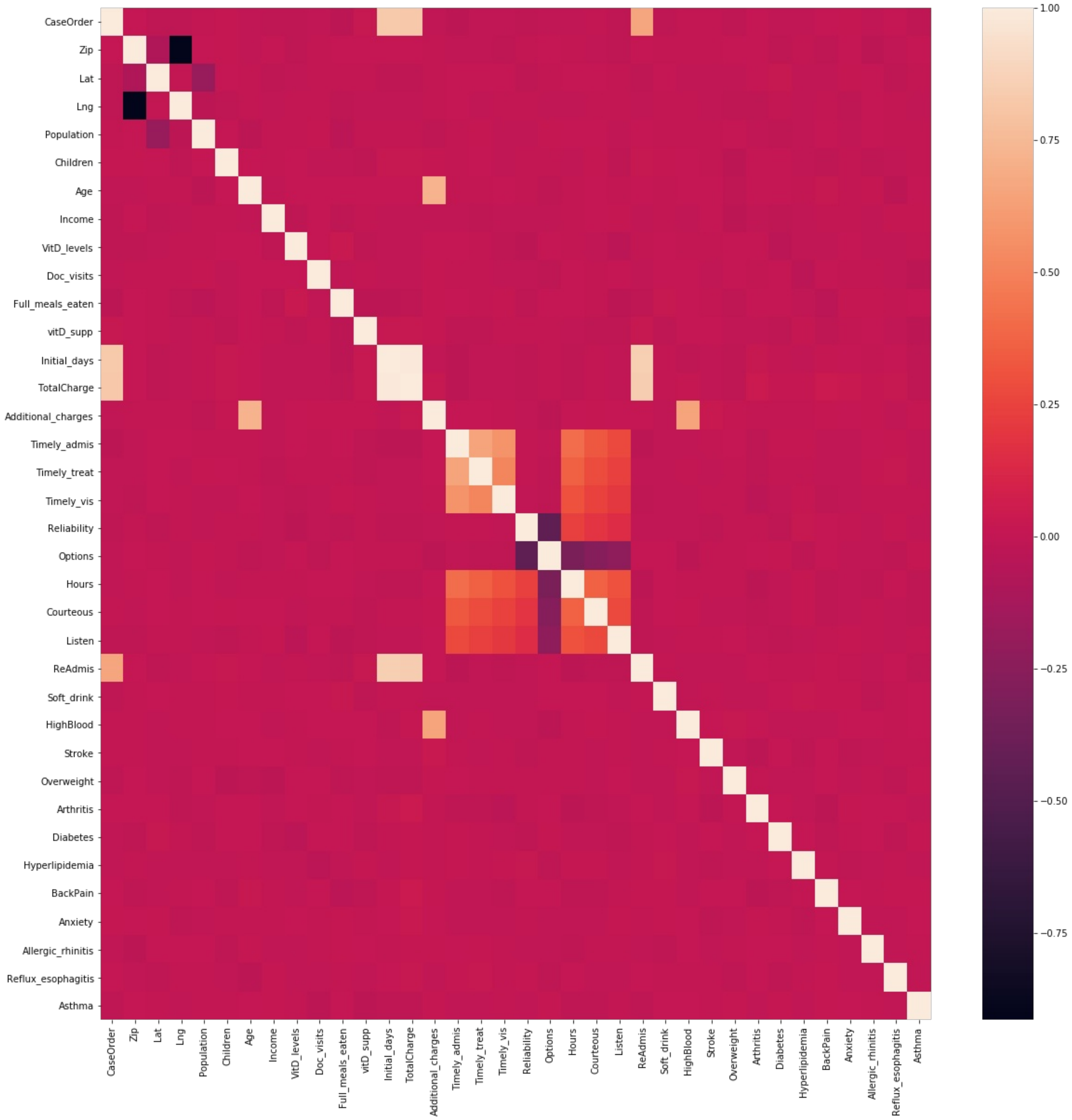
Out[38]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	VitD_levels	Doc_visits	...	Str
CaseOrder	1.000000	0.010465	-0.012946	-0.012081	0.001489	0.017027	-0.003011	-0.012265	-0.015026	-0.006920	...	0.000
Zip	0.010465	1.000000	-0.084258	-0.913573	0.012947	0.014307	-0.003327	0.010507	-0.010747	0.000257	...	0.004
Lat	-0.012946	-0.084258	1.000000	0.001062	-0.187334	0.005874	-0.000132	-0.015414	-0.005158	0.004689	...	-0.001
Lng	-0.012081	-0.913573	0.001062	1.000000	-0.018263	-0.014141	0.002780	-0.008175	0.000931	0.002417	...	-0.007
Population	0.001489	0.012947	-0.187334	-0.018263	1.000000	0.007810	-0.018884	0.002162	0.004719	0.016088	...	0.005
Children	0.017027	0.014307	0.005874	-0.014141	0.007810	1.000000	0.006050	0.003951	0.006542	-0.003467	...	-0.000
Age	-0.003011	-0.003327	-0.000132	0.002780	-0.018884	0.006050	1.000000	-0.003218	0.008795	0.010819	...	0.011
Income	-0.012265	0.010507	-0.015414	-0.008175	0.002162	0.003951	-0.003218	1.000000	-0.015684	0.011179	...	0.007
VitD_levels	-0.015026	-0.010747	-0.005158	0.000931	0.004719	0.006542	0.008795	-0.015684	1.000000	0.010297	...	0.002
Doc_visits	-0.006920	0.000257	0.004689	0.002417	0.016088	-0.003467	0.010819	0.011179	0.010297	1.000000	...	-0.005
Full_meals_eaten	-0.020805	0.013077	-0.001353	-0.013120	-0.025711	-0.005112	0.008499	-0.012628	0.032606	-0.004586	...	0.002
vitD_supp	0.026011	0.009348	0.005225	-0.001817	0.004134	-0.010125	0.009336	0.001478	-0.015671	0.002755	...	0.006
Initial_days	0.831426	0.011103	-0.009938	-0.006659	0.004435	0.022122	0.009943	-0.006543	-0.007267	-0.008363	...	-0.006
TotalCharge	0.821397	0.010493	-0.012843	-0.005866	0.004758	0.022909	0.010785	-0.008523	-0.004403	-0.005363	...	-0.007
Additional_charges	-0.003178	0.001545	-0.001433	0.003290	-0.011835	0.014076	0.716409	-0.005190	0.006120	0.014611	...	0.033
Timely_admis	-0.016607	-0.008630	0.008075	0.011933	0.004194	0.004097	0.005614	-0.004194	0.010499	0.003984	...	0.002
Timely_treat	-0.005508	-0.002475	0.009184	-0.002521	0.016837	0.006169	0.004382	-0.012371	0.003697	0.004377	...	-0.008
Timely_vis	-0.006320	-0.010277	0.010924	0.002614	-0.004754	-0.002485	0.006990	-0.007394	-0.011930	-0.003794	...	0.002
Reliability	-0.016204	0.001231	-0.011577	0.000283	-0.008892	-0.001091	0.003407	-0.003532	-0.016650	-0.006303	...	-0.013
Options	-0.004709	0.006290	0.000179	-0.002771	0.013720	0.003409	-0.013980	-0.005088	0.007878	-0.011124	...	0.004
Hours	-0.006087	-0.001406	0.009542	-0.004637	0.007970	-0.002796	0.003434	0.003083	0.004610	0.009226	...	-0.000
Courteous	0.005102	-0.004203	0.009071	0.002070	0.010529	0.015894	0.009339	0.008516	-0.007461	0.005322	...	-0.001
Listen	-0.012319	-0.010159	0.004348	0.003871	-0.005522	-0.011509	0.002873	0.020238	-0.024347	0.006145	...	0.001
ReAdmis	0.661462	0.009519	-0.012324	-0.004241	0.007563	0.023890	0.011880	-0.008669	0.002858	-0.002226	...	-0.004
Soft_drink	-0.014014	-0.000717	0.010164	-0.000521	0.001137	0.017635	0.000994	0.001186	0.009452	0.013203	...	0.004
HighBlood	0.005010	0.004015	-0.003431	0.002898	-0.000321	0.006837	0.008265	-0.003876	0.004668	0.012391	...	0.005
Stroke	0.000329	0.004496	-0.001691	-0.007993	0.005435	-0.000677	0.011657	0.007790	0.002774	-0.005126	...	1.000
Overweight	-0.015372	0.008915	-0.003979	-0.012112	0.007289	-0.017994	-0.009858	-0.018495	0.010462	0.010228	...	-0.002
Arthritis	0.008205	0.010427	0.011206	-0.014313	-0.006209	0.008645	0.006977	-0.006684	0.006214	0.000778	...	-0.017
Diabetes	-0.006454	-0.009222	0.022006	0.004272	-0.009324	0.012099	0.007367	-0.010296	-0.024080	0.012104	...	0.011
Hyperlipidemia	-0.006078	0.004892	-0.003521	-0.004862	-0.003931	-0.001939	0.004963	-0.003735	-0.008225	-0.026669	...	-0.009
BackPain	0.012056	-0.010125	-0.008733	-0.000215	0.011799	-0.013339	0.026304	0.003499	-0.008061	0.012149	...	0.007
Anxiety	0.014668	0.007898	0.015698	-0.012313	-0.002665	0.002848	0.001124	0.003670	0.006645	0.002448	...	-0.010
Allergic_rhinitis	-0.001750	-0.017582	0.013252	0.012777	0.008393	-0.015710	0.013334	-0.004113	-0.005865	0.001936	...	-0.003
Reflux_esophagitis	0.010788	-0.002543	-0.013755	0.000970	0.004870	-0.007773	-0.018236	0.014447	-0.008863	-0.006894	...	0.004
Asthma	-0.015245	0.010517	0.000171	-0.006567	0.002971	0.002161	0.011092	0.013542	0.000382	-0.016877	...	0.000

36 rows × 36 columns

In [39]:

```
fig_dims = (20, 20)
fig, ax = plt.subplots(figsize=fig_dims)
sns.heatmap(df.corr(), ax=ax)
plt.show()
```

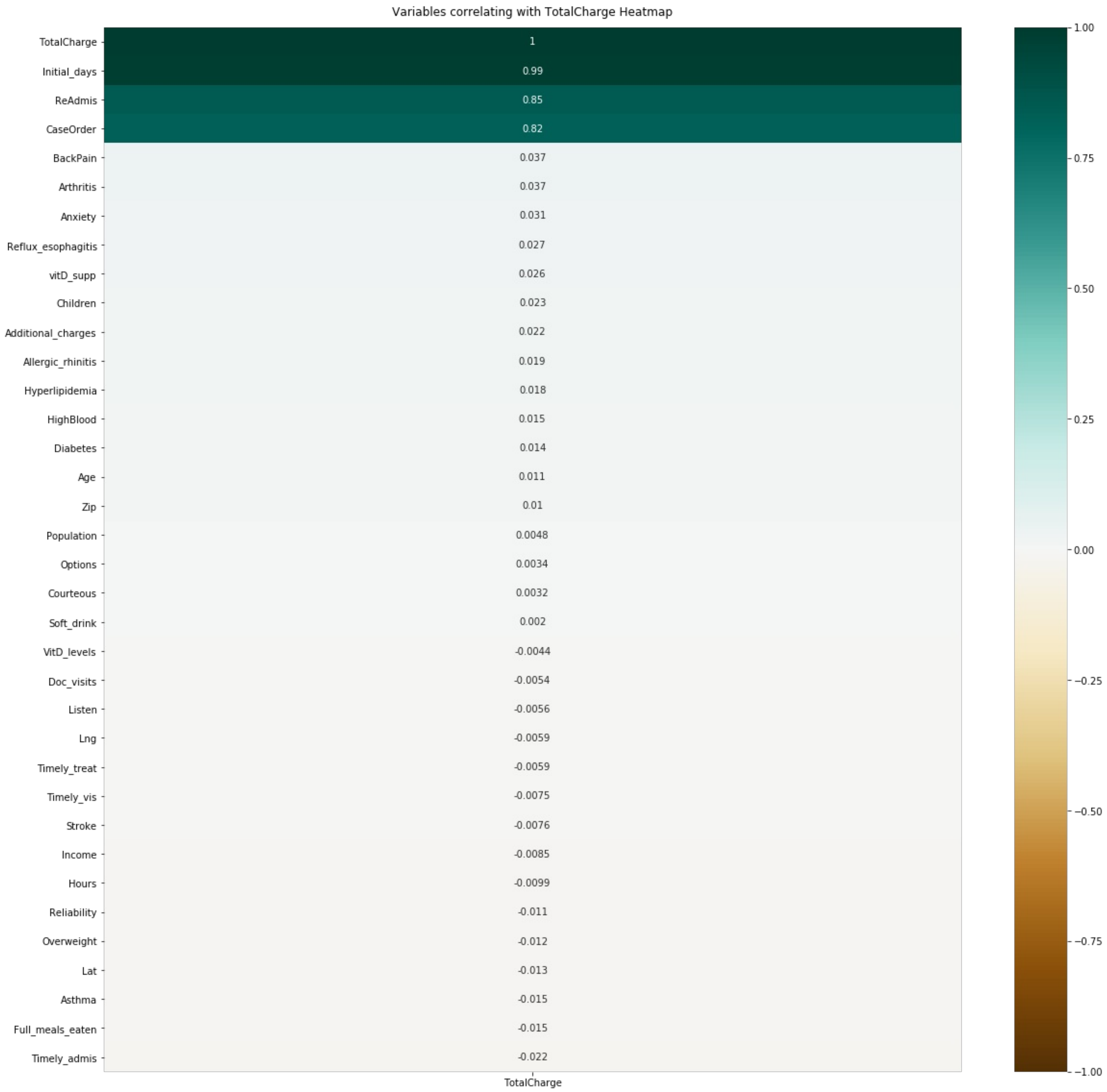


In [40]:

```
# Heatmap code reference (Seaborn.heatmap, n.d.)

plt.figure(figsize=(20,20))
heatmap = sns.heatmap(df.corr()[['TotalCharge']].sort_values(by='TotalCharge', ascending=False), vmin=-1, vmax=
heatmap.set_title('Variables correlating with TotalCharge Heatmap',pad=12)
```

Out[40]:

```
Text(0.5, 1, 'Variables correlating with TotalCharge Heatmap')
```



Step 5 : Create dummy variables (ensure n = k number of varibles) & rename any necessary features

In [41]:
```
#Get dummies code reference (Pandas.get_dummies, N.d.)

df = pd.get_dummies(df, columns=['ReAdmis','Children','Marital','Gender','Doc_visits','vitD_supp','Soft_drink',
```

In [42]:
```
df.rename(columns={'Services_CT Scan':'Services_CT_Scan','Marital_Never Married':'Marital_Never_Married','Initi
```

Step 6 drop demographic features that won't be used in the analysis

In [43]:
```
df.drop(['CaseOrder','Customer_id','Interaction','UID','City','State','County','Area','TimeZone','Job','Lng','L
```

Step 7 select features for regression statistically - abs > .04 correlation with TotalCharge

In [44]:
```
abs(df.corr()["TotalCharge"][abs(df.corr()["TotalCharge"])>0.04].drop('TotalCharge')).index.tolist()
```

Out[44]:
```
['Initial_days',
 'ReAdmis_0',
 'ReAdmis_1',
 'Initial_admin_Elective_Admission',
 'Initial_admin_Emergency_Admission',
 'Initial_admin_Observation_Admission',
 'Complication_risk_High',
 'Complication_risk_Medium']
```

In [45]:
```
df = df[['TotalCharge','Initial_days',
 'ReAdmis_0',
 'ReAdmis_1',
 'Initial_admin_Elective_Admission',
 'Initial_admin_Emergency_Admission',
 'Initial_admin_Observation_Admission',
 'Complication_risk_High',
 'Complication_risk_Medium']]
df
```

Out[45]:

	TotalCharge	Initial_days	ReAdmis_0	ReAdmis_1	Initial_admin_Elective_Admission	Initial_admin_Emergency_Admission	Initial_admin_C
0	3726.702860	10.585770	1	0	0	1	
1	4193.190458	15.129562	1	0	0	1	
2	2434.234222	4.772177	1	0	1	0	
3	2127.830423	1.714879	1	0	1	0	
4	2113.073274	1.254807	1	0	1	0	
...	...	...	...	...	...	...	
9995	6850.942000	51.561220	1	0	0	1	
9996	7741.690000	68.668240	0	1	1	0	
9997	8276.481000	70.154180	0	1	1	0	
9998	7644.483000	63.356900	0	1	0	1	
9999	7887.553000	70.850590	0	1	0	0	

9206 rows × 9 columns

Step 8 Min-Max scaling of features

In [ ]:

In [46]:
```
df = (df - df.min()) / (df.max() - df.min())
df
```

Out[46]:

	TotalCharge	Initial_days	ReAdmis_0	ReAdmis_1	Initial_admin_Elective_Admission	Initial_admin_Emergency_Admission	Initial_admin_C
0	0.246933	0.135022	1.0	0.0	0.0	1.0	
1	0.311343	0.199037	1.0	0.0	0.0	1.0	
2	0.068475	0.053117	1.0	0.0	1.0	0.0	
3	0.026168	0.010044	1.0	0.0	1.0	0.0	
4	0.024130	0.003562	1.0	0.0	1.0	0.0	
...	...	...	...	...	...	...	
9995	0.678314	0.712308	1.0	0.0	0.0	1.0	
9996	0.801304	0.953321	0.0	1.0	1.0	0.0	
9997	0.875146	0.974256	0.0	1.0	1.0	0.0	
9998	0.787882	0.878492	0.0	1.0	0.0	1.0	
9999	0.821444	0.984067	0.0	1.0	0.0	0.0	

9206 rows × 9 columns

In [47]:
```
#Read out prepared data set for submission.
```

In [48]:
```
df.to_excel('C:/Users/ericy/Desktop/d209.1_prepared.xlsx')
```

In [ ]:

Step 9 Summary stats for selected features and TotalCharge

In [49]:
```
df.isnull().sum()
```

Out[49]:
```
TotalCharge                          0
Initial_days                         0
ReAdmis_0                            0
ReAdmis_1                            0
Initial_admin_Elective_Admission     0
Initial_admin_Emergency_Admission    0
Initial_admin_Observation_Admission  0
Complication_risk_High               0
Complication_risk_Medium             0
dtype: int64
```

In [50]:
```
df.describe()
```

Out[50]:

	TotalCharge	Initial_days	ReAdmis_0	ReAdmis_1	Initial_admin_Elective_Admission	Initial_admin_Emergency_Admission	Initial_admi
count	9206.000000	9206.000000	9206.000000	9206.000000	9206.000000	9206.000000	
mean	0.465055	0.470530	0.633065	0.366935	0.250489	0.505323	
std	0.301177	0.370886	0.481995	0.481995	0.433318	0.499999	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.171211	0.096921	0.000000	0.000000	0.000000	0.000000	
50%	0.436588	0.420396	1.000000	0.000000	0.000000	1.000000	
75%	0.762208	0.847510	1.000000	1.000000	1.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

In [51]:
```
df.corr()
```

Out[51]:

	TotalCharge	Initial_days	ReAdmis_0	ReAdmis_1	Initial_admin_Elective_Admission	Initial_admin_Emerge
TotalCharge	1.000000	0.987666	-0.845034	0.845034	-0.055349	
Initial_days	0.987666	1.000000	-0.852064	0.852064	0.011339	
ReAdmis_0	-0.845034	-0.852064	1.000000	-1.000000	0.010482	
ReAdmis_1	0.845034	0.852064	-1.000000	1.000000	-0.010482	
Initial_admin_Elective_Admission	-0.055349	0.011339	0.010482	-0.010482	1.000000	
Initial_admin_Emergency_Admission	0.107284	-0.010895	-0.019393	0.019393	-0.584290	
Initial_admin_Observation_Admission	-0.069032	0.001243	0.011997	-0.011997	-0.328595	
Complication_risk_High	0.081082	-0.008235	0.004294	-0.004294	0.022966	
Complication_risk_Medium	-0.064676	-0.006597	-0.006905	0.006905	-0.015254	

In [52]:
```
df.mean()
```

Out[52]:
```
TotalCharge                          0.465055
Initial_days                         0.470530
ReAdmis_0                            0.633065
ReAdmis_1                            0.366935
Initial_admin_Elective_Admission     0.250489
Initial_admin_Emergency_Admission    0.505323
Initial_admin_Observation_Admission  0.244189
Complication_risk_High               0.335108
Complication_risk_Medium             0.452857
dtype: float64
```

In [53]:
```
df.median()
```

Out[53]:
```
TotalCharge                          0.436588
Initial_days                         0.420396
ReAdmis_0                            1.000000
ReAdmis_1                            0.000000
Initial_admin_Elective_Admission     0.000000
Initial_admin_Emergency_Admission    1.000000
Initial_admin_Observation_Admission  0.000000
Complication_risk_High               0.000000
Complication_risk_Medium             0.000000
dtype: float64
```

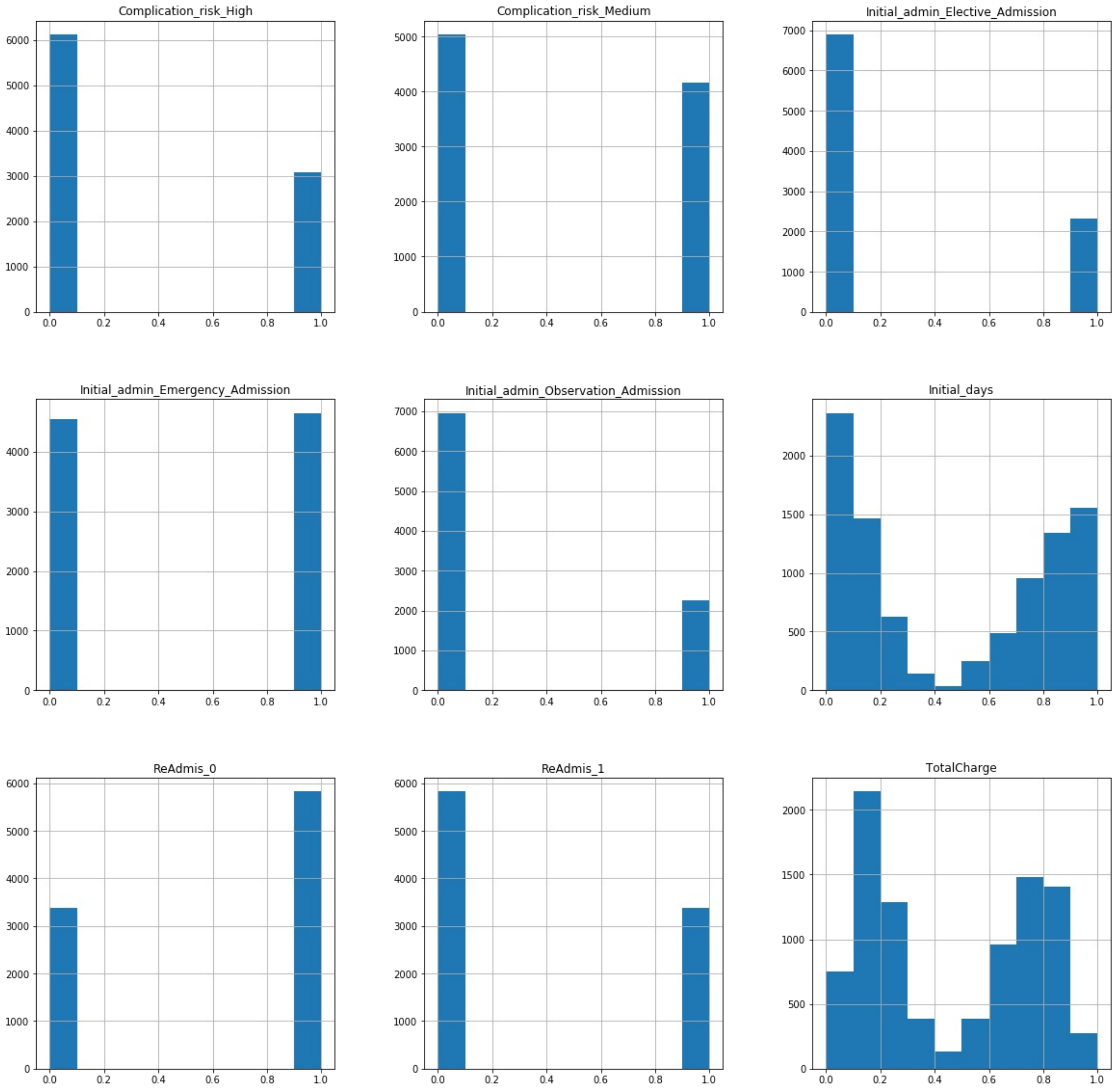
In [54]:
```
df.mode()
```

Out[54]:

	TotalCharge	Initial_days	ReAdmis_0	ReAdmis_1	Initial_admin_Elective_Admission	Initial_admin_Emergency_Admission	Initial_admin_Obse
0	0.775589	0.935755	1.0	0.0	0.0	1.0	
1	0.832094	0.976668	NaN	NaN	NaN	NaN	

In [55]:
```
df.hist(figsize=(20,20))
```

Out[55]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001925FC95E88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000019261986B48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000019261B6EB48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000192620A2C88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000192623ABD88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000192623B4EC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000019261A87A88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000192622C3FC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000192622B7B48>]],
      dtype=object)
```



In [ ]:

Step 10 .astype() categorical features to int8

In [56]:
```
df.astype({'ReAdmis_0': 'int8','ReAdmis_1':'int8','Initial_admin_Elective_Admission':'int8','Initial_admin_Emer
```

Out[56]:
```
TotalCharge                              float64
Initial_days                             float64
ReAdmis_0                                   int8
ReAdmis_1                                   int8
Initial_admin_Elective_Admission            int8
Initial_admin_Emergency_Admission           int8
Initial_admin_Observation_Admission         int8
Complication_risk_High                      int8
Complication_risk_Medium                    int8
dtype: object
```

In [ ]:

Step 11 Assign prepared features to y = TotalCharge, X = Prepared independent features.

In [57]:
```
y=df['TotalCharge']
X = df.drop(columns = 'TotalCharge')
```

Step 12 import necessary libraries for Decision Tree regression analysis and cross validation

split data into test and train sets.

In [ ]:

In [58]:
```
from sklearn import tree
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error as MSE
```

Section D Data Analysis

In [59]:
```
#Instantiate the model
# Code reference (Instantiate the model, n.d.)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = .7, test_size = 0.30, random_state = 1)
```

In [60]:
```
X_train.to_excel('C:/Users/ericy/Desktop/d209.1.X_train.xlsx')
```

In [61]:
```
X_test.to_excel('C:/Users/ericy/Desktop/d209.1.X_test.xlsx')
```

In [62]:
```
y_train.to_excel('C:/Users/ericy/Desktop/d209.1.y_train.xlsx')
```

In [63]:
```
y_test.to_excel('C:/Users/ericy/Desktop/d209.1.y_test.xlsx')
```

In [64]:
```
#Instantiate Decision Tree

regr_1 = tree.DecisionTreeRegressor(max_depth=3, min_samples_leaf=.1,random_state=1)
regr_1.fit(X_train,y_train)
```

Out[64]:
```
DecisionTreeRegressor(max_depth=3, min_samples_leaf=0.1, random_state=1)
```

In [65]:
```
# Predict

y_pred = regr_1.predict(X_test)
```

Model Accuracy

In [66]:
```
# Accuracy of the model
accuracy = regr_1.score(X_test, y_test)
print(accuracy)
```

```
0.9618135955188434
```

In [67]:
```
#Read predictions and actual y values to variable

ap = pd.DataFrame(data={'Predicted': y_pred, 'Actual': y_test}).head(3000)
ap
```

Out[67]:

	Predicted	Actual
5078	0.771395	0.767078
8973	0.647140	0.574817
9724	0.853578	0.774836
4849	0.108177	0.048150
5896	0.771395	0.743521
...	...	...
8514	0.647140	0.536069
1704	0.175889	0.134419
7034	0.647140	0.686924
3942	0.108177	0.039914
8391	0.647140	0.723192

2762 rows × 2 columns

In [68]:

```
# Calculate Mean Square Error

mse_dt = MSE(y_test, y_pred)
print('MSE: {:.4f}'.format(mse_dt))
```

```
MSE: 0.0034
```

In [69]:

```
# Compute the array containing the 10-folds CV MSEs
# Code Reference (Evaluate the 10-fold CV error, n.d.)

MSE_CV_scores = - cross_val_score(regr_1, X_train, y_train, cv=10,
                                  scoring='neg_mean_squared_error',
                                  n_jobs=-1)
```

In [70]:

```
# Compute Root Mean Square Error from cross val scores

RMSE_CV = (MSE_CV_scores.mean())**(1/2)

# Print RMSE_CV
print('CV RMSE: {:.4f}'.format(RMSE_CV))
```

```
CV RMSE: 0.0578
```

In [71]:

```
# Compute Root Mean Square Error from training set & predictions
# code reference (Evaluate the training error, n.d.)

# Fit dt to the training set
regr_1.fit(X_train, y_train)

# Predict the labels of the training set
y_pred_train = regr_1.predict(X_train)

# Evaluate the training set RMSE of regr_1
RMSE_train = (MSE(y_train, y_pred_train))**(1/2)

# Print RMSE_train
print('Train RMSE: {:.4f}'.format(RMSE_train))
```

```
Train RMSE: 0.0583
```

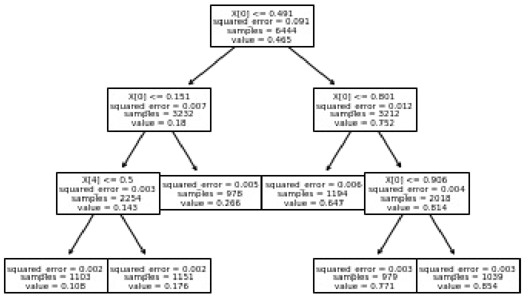
In [ ]:

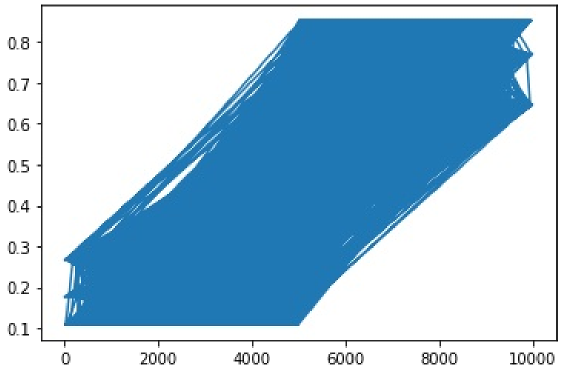
In [72]:

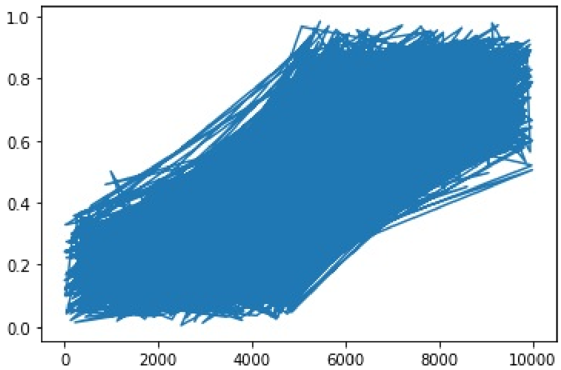
```
# Graph decision tree
# Code reference (Galarnyk, 2022)
tree.plot_tree(regr_1)
```

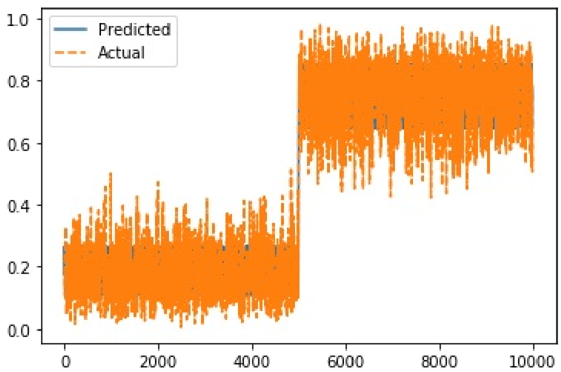
Out[72]:

```
[Text(0.5, 0.875, 'X[0] <= 0.491\nsquared_error = 0.091\nsamples = 6444\nvalue = 0.465'),
 Text(0.3, 0.625, 'X[0] <= 0.151\nsquared_error = 0.007\nsamples = 3232\nvalue = 0.18'),
 Text(0.2, 0.375, 'X[4] <= 0.5\nsquared_error = 0.003\nsamples = 2254\nvalue = 0.143'),
 Text(0.1, 0.125, 'squared_error = 0.002\nsamples = 1103\nvalue = 0.108'),
 Text(0.3, 0.125, 'squared_error = 0.002\nsamples = 1151\nvalue = 0.176'),
 Text(0.4, 0.375, 'squared_error = 0.005\nsamples = 978\nvalue = 0.266'),
 Text(0.7, 0.625, 'X[0] <= 0.801\nsquared_error = 0.012\nsamples = 3212\nvalue = 0.752'),
 Text(0.6, 0.375, 'squared_error = 0.006\nsamples = 1194\nvalue = 0.647'),
 Text(0.8, 0.375, 'X[0] <= 0.906\nsquared_error = 0.004\nsamples = 2018\nvalue = 0.814'),
 Text(0.7, 0.125, 'squared_error = 0.003\nsamples = 979\nvalue = 0.771'),
 Text(0.9, 0.125, 'squared_error = 0.003\nsamples = 1039\nvalue = 0.854')]
```

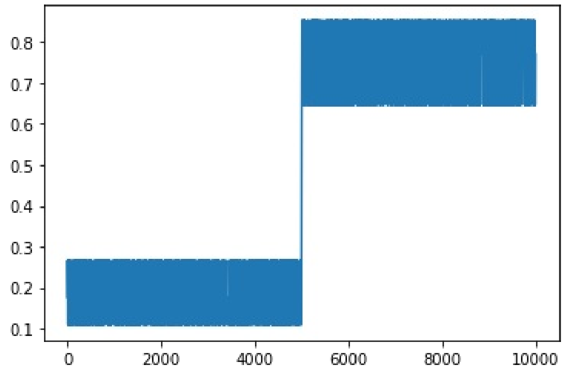


In [73]:
```
plt.plot(ap['Predicted'])
```
Out[73]:
```
[<matplotlib.lines.Line2D at 0x1926234eac8>]
```


In [74]:
```
plt.plot(ap['Actual'])
```
Out[74]:
```
[<matplotlib.lines.Line2D at 0x19261fe8ac8>]
```


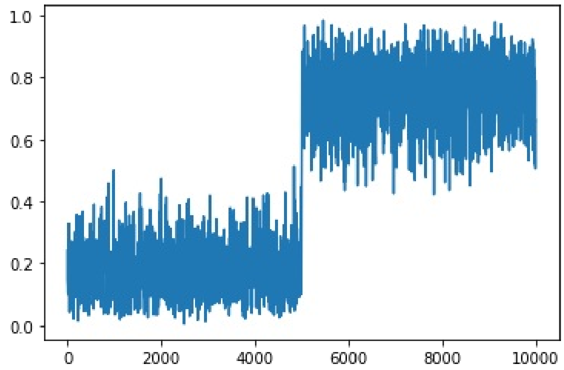
In [75]:
```
sns.lineplot(data=ap)
```
Out[75]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19261f944c8>
```


In [76]:
```
sns.lineplot(data=ap['Predicted'])
```
Out[76]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19261fca8c8>
```



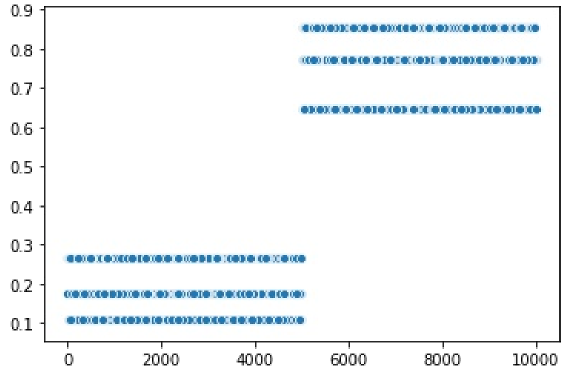
In [77]:
```
sns.lineplot(data=ap['Actual'])
```

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x192620703c8>

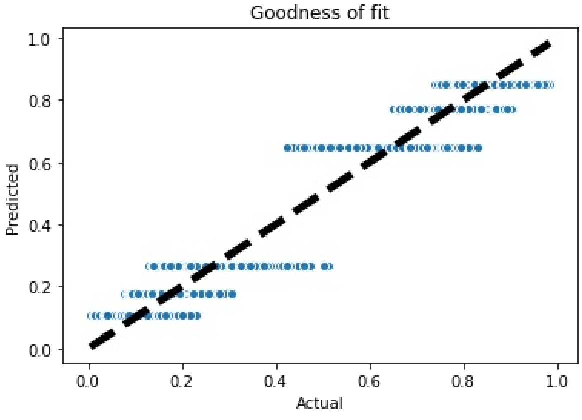


In [78]:
```
sns.scatterplot(data=ap['Predicted'])
```

Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x19262151ac8>



In [79]:
```
# Goodness of fit test, predicted vs actual

fig, ax = plt.subplots()

ax.scatter(y_test, y_pred, edgecolors=(1, 1, 1))
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=5)

ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Goodness of fit")
plt.show()
```



In [ ]: