D208

January 3, 2022

1 D208 Task 1 Multiple Regression for Predictive Modeling

1.1 Malcolm Mikkelsen

1.2 MSDA

1.3 D208: Predictive Modeling

1.4 28 November, 2021

2 Part 1: Research Question

2.1 Summarize a Research Question and Define Objectives of Analysis

The research question we will investigate is: using multiple continuous and categorical variables, can we accurately predict the cost to the patient of their initial admission? The goal of this analysis will be to allow stakeholders to examine variables that impact the cost of admission, which may increase the likelihood of a patient leaving before their condition has fully improved.

3 Part 2: Method Justification

3.1 Assumptions of Multiple Regression Models

The first assumption of any linear regression, including multiple regression models, is that the relationship between the dependent and independent variables is linear. Next, we assume that the independent variables are not too closely related between one another. The final two assumptions are that the residuals are normally distributed, and that the data is randomly selected from the population (Shin, 2021).

3.2 Benefits of Python Programming Language

I have selected Python for several reasons. It is one of the most highly utilized programming languages within analytics, meaning that there is an abundance of resources for it online and in text. It is a flexible coding language that allows for us to import several beneficial packages in, including numpy, scikit, and pandas which are used during this project (Larose, 2019).

3.3 Why Use Multiple Regression Modeling

By using multiple regression modeling techniques, we can better account for a multitude of factors impacting our outcome we desire. Instead of assuming that there is only one independent variable to manipulate, we can get a fuller picture of what is impacting the outcome. This is beneficial, because as we often see in business, it is rarely just one factor. Multiple regression is also a preferred method when our output, or dependent variable, is a continuous variable.

4 Part 3: Data Preparation

4.1 Describe Techniques

We will clean the data using the following steps. First, we will identify null values and outliers and address them through multiple methods including counting the number of null values as described by the Big Data Zone (Needham, 2019). We will then replace null values by inputting the mean value of continuous variables. We will assume for categorical questions, that the patient not responding means the question did not apply to them meaning we will input no as their answer. Then we will drop data that presents as an unexplainable outlier. For cost of stay we will allow the outliers to remain due to the variability in cost depending on what procedures they had done during their stay. Then we will recast all categorical data using dictionaries. For yes/no binary variables to be 0 for no and 1 for yes. (Pandas Development Team, 2008)

In order to improve speed of analysis and decrease the possibility of utilizing the wrong data, I will create a new Data Frame named desired medical data(dmd) to perform the regression modeling on.

4.2 Summary Statistics and Choosing Predictors

When performing the analysis, we will be looking at a large number of predictor variables. The variables were chosen based on whether they could logically require treatment that may increase cost of admission. The continuous variables include days of initial admission, income, age, number of full meals eaten, and the number of doctor visits. The categorical variables are gender, whether Vitamin D supplementation was administered, whether the following conditions were present: high blood pressure, previous stroke, overweight, arthritis, diabetes, hyperlipidemia, back pain, anxiety, allergic rhinitis, asthma, and reflux esophagitis.

4.3 Univariate and Bivariate Visualizations

I will generate a histogram for each predictor to show the distribution of each. For the bivariate statistics I will only be creating visualizations for the continuous variables. This is because categorical data will present lines of points at each of the possible answers, and do not provide any insight into the data.

4.4 Save Cleaned Data

Below is the code for importing the libraries, creating the new dataframe, identifying and addressing outliers, and for creating the visualizations of univariate and bivariate statistics. Lastly the data file will be saved as clean_data_208 and will be uploaded with the submission.

5 Data Cleaning

[1]:
```python
# Load in data set and libraries
import pandas as pd
import numpy as np
import seaborn as sns


md = pd.read_csv('medical_raw_data.csv')
#md.columns.tolist()
md.dtypes
```

[1]:
```
Unnamed: 0              int64
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              float64
Age                   float64
Education              object
Employment             object
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            float64
Arthritis              object
Diabetes               object
```

```
Hyperlipidemia          object
BackPain                object
Anxiety                float64
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
```

[2]:
```
#create new dataframe with needed columns
dmd = md[['Age', 'Income', 'Gender', 'Doc_visits', 'Full_meals_eaten',␣
↪'VitD_supp', 'HighBlood', 'Stroke',
         'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',␣
↪'Hyperlipidemia', 'BackPain', 'Anxiety',
         'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Initial_days',␣
↪'TotalCharge', 'VitD_levels']].copy()
```

[3]:
```
# Recast Gender
dict_gender = {'Gender':{'Male': 1, 'Female': 2, 'Prefer not to answer':'3'}}
dmd.replace(dict_gender, inplace=True)
```

[4]:
```
#High blood pressure
dict_bp = {'HighBlood':{ 'Yes' : 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_bp, inplace = True)
```

[5]:
```
#Stroke
dict_stroke = {'Stroke':{ 'Yes' : 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_stroke, inplace = True)
```

[6]:
```
#Complication Risk
dict_comp = {'Complication_risk':{'Low': 1, 'Medium': 2, 'High': 3}}
dmd.replace(dict_comp, inplace = True)
```

[7]:
```
# Athritis
dict_ath = {'Arthritis':{'Yes' : 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_ath, inplace = True)
```

[8]:
```
#Diabetes
dict_dia = {'Diabetes':{'Yes' : 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_dia, inplace = True)
```

[9]:
```
#Hyperlipidemia
dict_hype = {'Hyperlipidemia':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_hype, inplace = True)
```

[10]:
```
#Back Pain
dict_back = {'BackPain':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_back, inplace = True)
```

[11]:
```
#Allergic Rhinitis
dict_aller = {'Allergic_rhinitis':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_aller, inplace = True)
```

[12]:
```
# Reflux Esophagitis
dict_ref = {'Reflux_esophagitis':{'Yes': 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_ref, inplace = True)
```

[13]:
```
#Asthma
dict_ast = {'Asthma': {'Yes': 1, 'No': 0, 'NA': np.NaN}}
dmd.replace(dict_ast, inplace = True)
```

5.1 Null Value Check and Correction

[14]:
```
# Verify that there are no null values
dmd_null = dmd.isnull().sum()
print(dmd_null)
```

```
Age                   2414
Income                2464
Gender                   0
Doc_visits               0
Full_meals_eaten         0
VitD_supp                0
HighBlood                0
Stroke                   0
Complication_risk        0
Overweight             982
Arthritis                0
Diabetes                 0
Hyperlipidemia           0
BackPain                 0
Anxiety                984
Allergic_rhinitis        0
Reflux_esophagitis       0
Asthma                   0
```

```
Initial_days          1056
TotalCharge              0
VitD_levels              0
dtype: int64
```

[15]:
```
# Replace categorical nulls with 0
dmd.Anxiety.fillna(0, inplace = True)
dmd.Overweight.fillna(0, inplace = True)

#Check to make sure the nulls were replaced
null_columns=dmd.columns[dmd.isnull().any()]
dmd[null_columns].isnull().sum()
```

[15]:
```
Age             2414
Income          2464
Initial_days    1056
dtype: int64
```

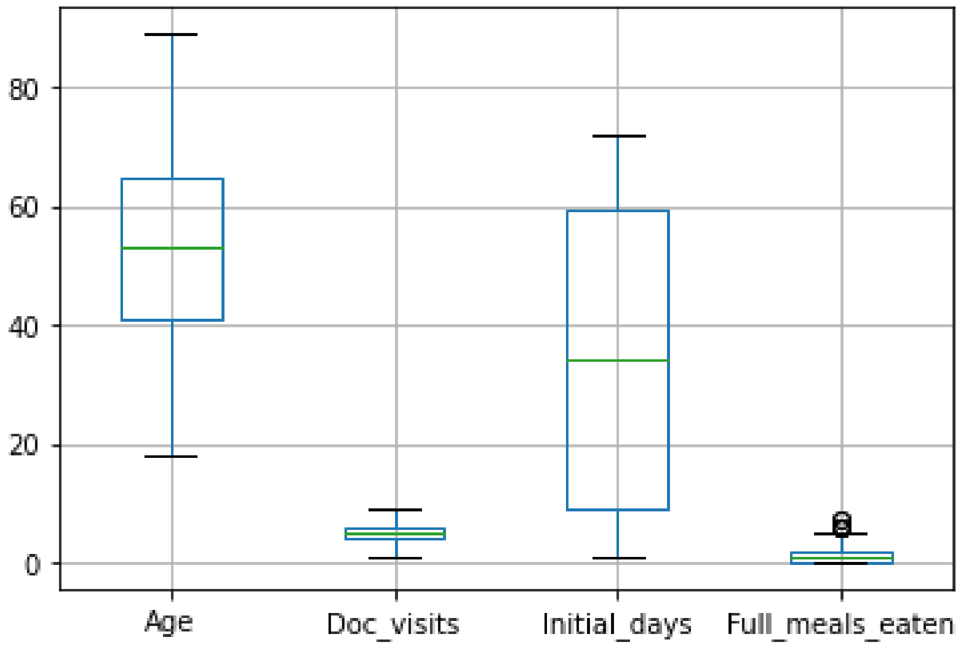
[16]:
```
#Replace remaining Null values with the mean value of the column
dmd['Age'] = dmd['Age'].fillna((dmd['Age'].mean()))
dmd['Initial_days'] = dmd['Initial_days'].fillna((dmd['Initial_days'].mean()))
dmd['Income'] = dmd['Income'].fillna((dmd['Income'].mean()))
#Check to make sure the nulls were replaced
null_columns=dmd.columns[dmd.isnull().any()]
dmd[null_columns].isnull().sum()
```

[16]:
```
Series([], dtype: float64)
```

6 Outlier Check

[17]:
```
#Using a boxplot to identify outliers in the rows most likely to contain them
dmd.boxplot(['Age', 'Doc_visits', 'Initial_days', 'Full_meals_eaten'])
```

[17]:
```
<AxesSubplot:>
```



6.1 Summary Statistics

[18]:
```
dmd.describe()
```

[18]:
```
                Age         Income     Doc_visits  Full_meals_eaten  \
count  10000.000000   10000.000000  10000.000000      10000.000000
mean      53.295676   40484.438268      5.012200          1.001400
std       17.993375   24883.598484      1.045734          1.008117
min       18.000000     154.080000      1.000000          0.000000
25%       41.000000   23956.162500      4.000000          0.000000
50%       53.295676   40484.438268      5.000000          1.000000
75%       65.000000   46466.797500      6.000000          2.000000
max       89.000000  207249.130000      9.000000          7.000000

           VitD_supp     HighBlood        Stroke  Complication_risk  \
count  10000.000000  10000.000000  10000.000000       10000.000000
mean       0.398900      0.409000      0.199300           2.123300
std        0.628505      0.491674      0.399494           0.730172
min        0.000000      0.000000      0.000000           1.000000
25%        0.000000      0.000000      0.000000           2.000000
50%        0.000000      0.000000      0.000000           2.000000
75%        1.000000      1.000000      0.000000           3.000000
max        5.000000      1.000000      1.000000           3.000000

         Overweight     Arthritis      Diabetes  Hyperlipidemia      BackPain  \
```

```
count  10000.00000  10000.000000  10000.00000    10000.000000  10000.000000
mean       0.63950       0.357400      0.27380        0.337200      0.411400
std        0.48017       0.479258      0.44593        0.472777      0.492112
min        0.00000       0.000000      0.00000        0.000000      0.000000
25%        0.00000       0.000000      0.00000        0.000000      0.000000
50%        1.00000       0.000000      0.00000        0.000000      0.000000
75%        1.00000       1.000000      1.00000        1.000000      1.000000
max        1.00000       1.000000      1.00000        1.000000      1.000000

            Anxiety  Allergic_rhinitis  Reflux_esophagitis        Asthma  \
count  10000.000000       10000.000000        10000.000000   10000.00000
mean       0.290600           0.394100            0.413500       0.28930
std        0.454062           0.488681            0.492486       0.45346
min        0.000000           0.000000            0.000000       0.00000
25%        0.000000           0.000000            0.000000       0.00000
50%        0.000000           0.000000            0.000000       0.00000
75%        1.000000           1.000000            1.000000       1.00000
max        1.000000           1.000000            1.000000       1.00000

       Initial_days   TotalCharge   VitD_levels
count  10000.000000  10000.000000  10000.000000
mean      34.432082   5891.538261     19.412675
std       24.860232   3377.558136      6.723277
min        1.001981   1256.751699      9.519012
25%        8.928987   3253.239465     16.513171
50%       34.432082   5852.250564     18.080560
75%       59.459981   7614.989701     19.789740
max       71.981486  21524.224210     53.019124
```

6.2 Univariate Statistics

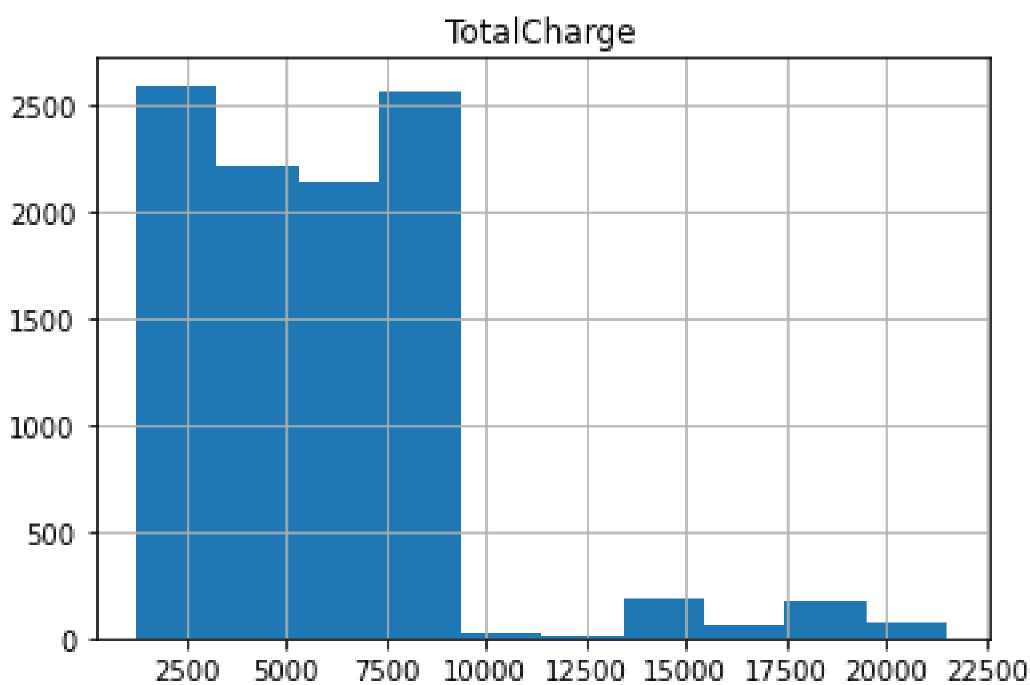
[19]:
```
#Run a histogram to see distribution of predictor variables
dmd[['TotalCharge']].hist()
dmd[['Doc_visits']].hist()
dmd[['Age']].hist()
dmd[['VitD_supp']].hist()
dmd[['Overweight']].hist()
dmd[['HighBlood']].hist()
dmd[['Diabetes']].hist()
dmd[['Hyperlipidemia']].hist()
dmd[['Age']].hist()
dmd[['Income']].hist()
dmd[['Doc_visits']].hist()
dmd[['Full_meals_eaten']].hist()
dmd[['Stroke']].hist()
dmd[['Complication_risk']].hist()
dmd[['Arthritis']].hist()
```

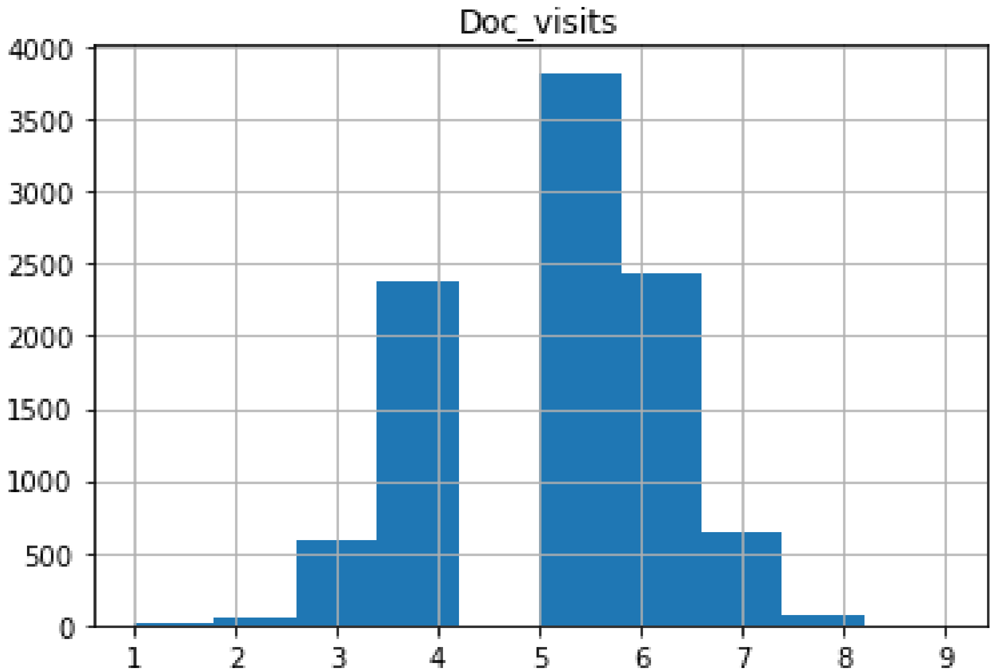
```
dmd[['BackPain']].hist()
dmd[['Anxiety']].hist()
dmd[['Allergic_rhinitis']].hist()
dmd[['Reflux_esophagitis']]
dmd[['Asthma']].hist()
dmd[['Initial_days']].hist()
dmd[['TotalCharge']].hist()
```

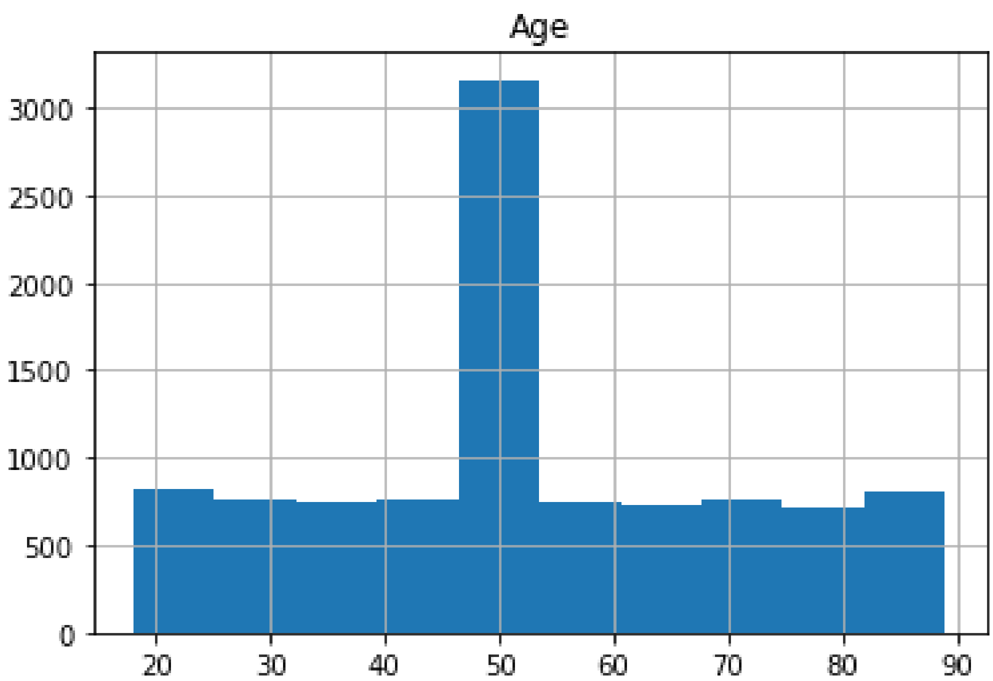
```
C:\Users\Mikke\Anaconda3\lib\site-
packages\pandas\plotting\_matplotlib\tools.py:196: RuntimeWarning: More than 20
figures have been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  fig = plt.figure(**fig_kw)
```

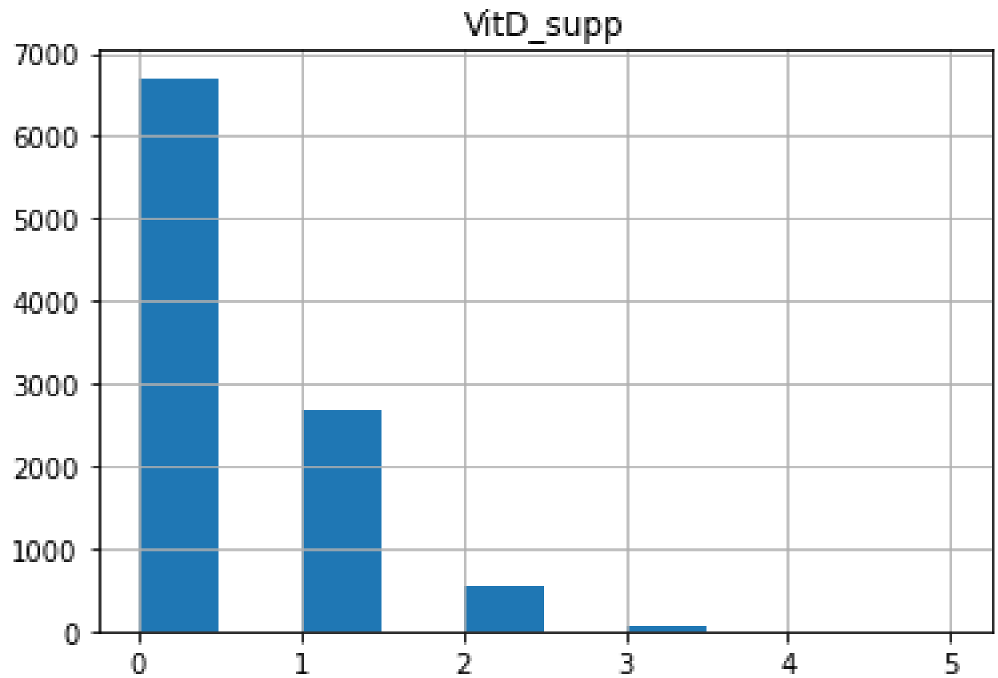
[19]:

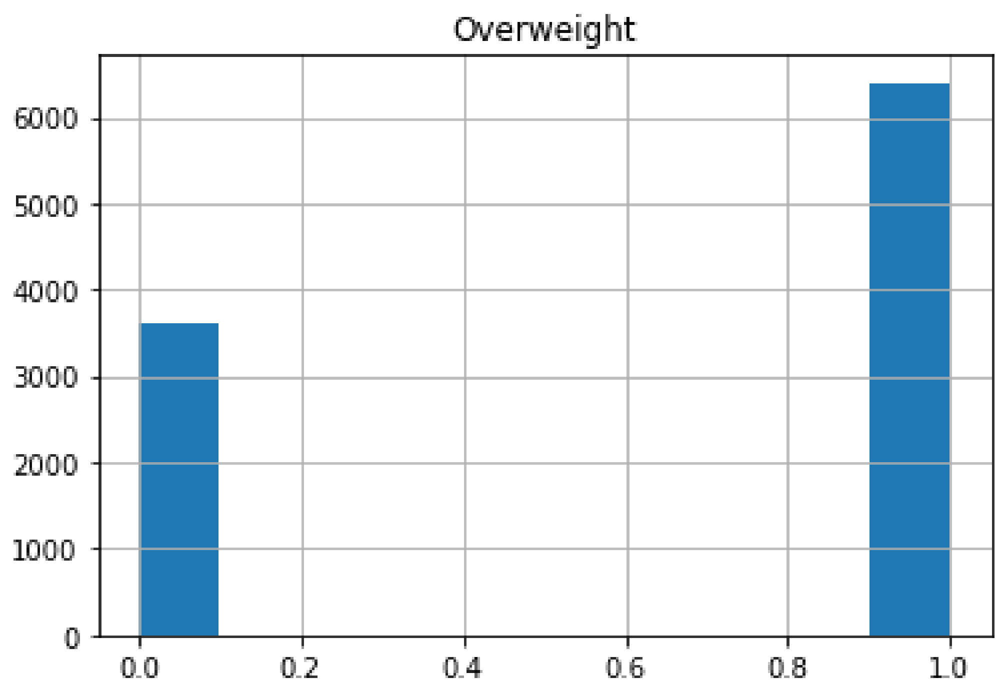
```
array([[<AxesSubplot:title={'center':'TotalCharge'}>]], dtype=object)
```

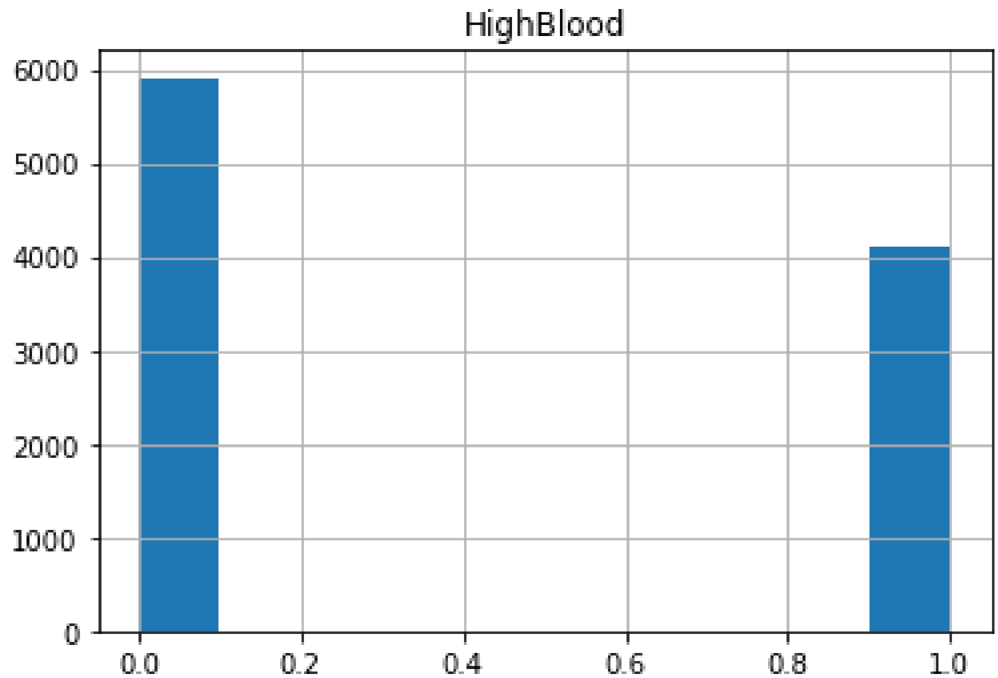


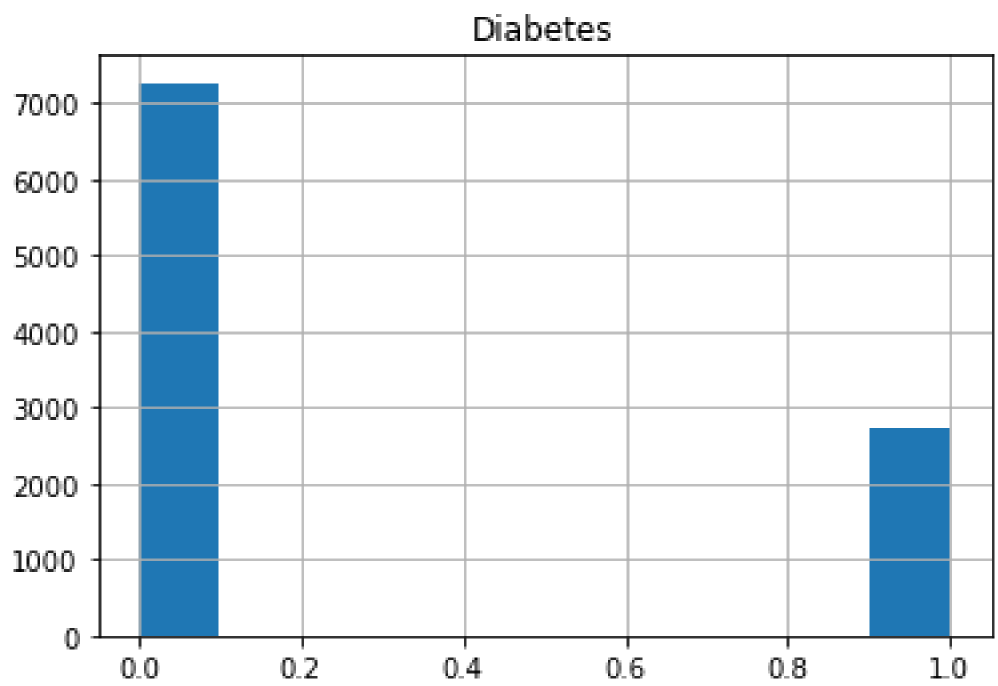

Doc_visits
0
500
1000
1500
2000
2500
3000
3500
4000
1
2
3
4
5
6
7
8
9

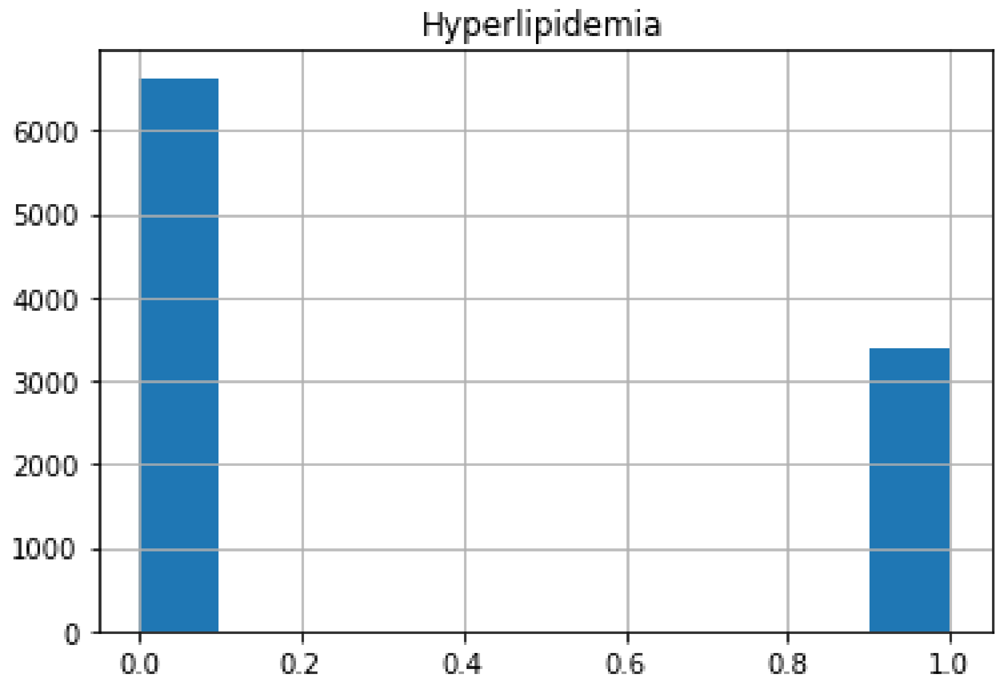

Age
0
500
1000
1500
2000
2500
3000
20
30
40
50
60
70
80
90

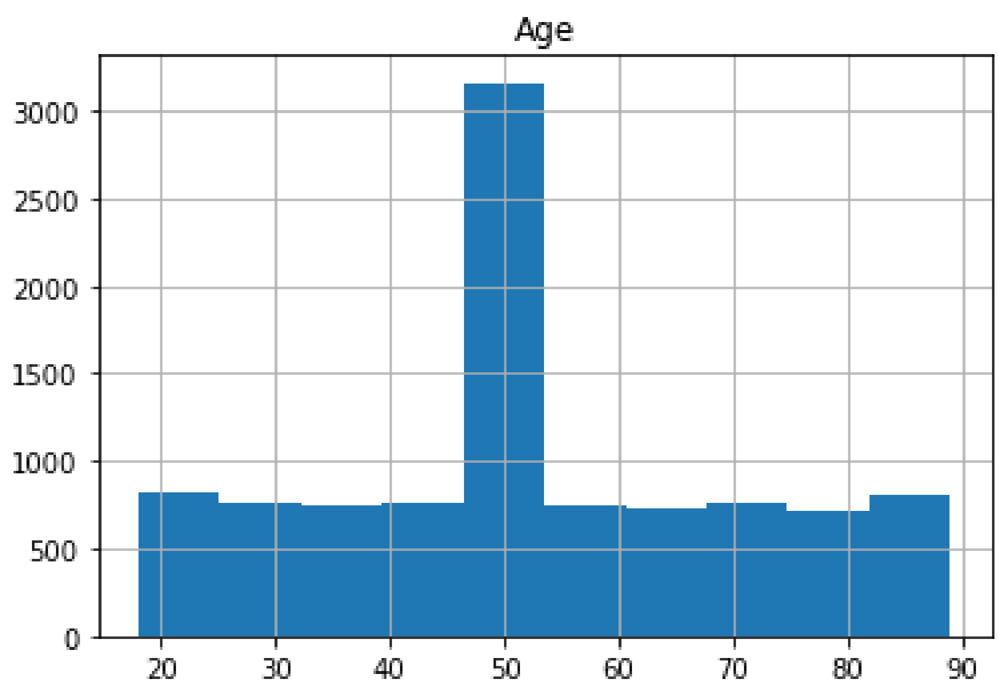

VitD_supp
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5

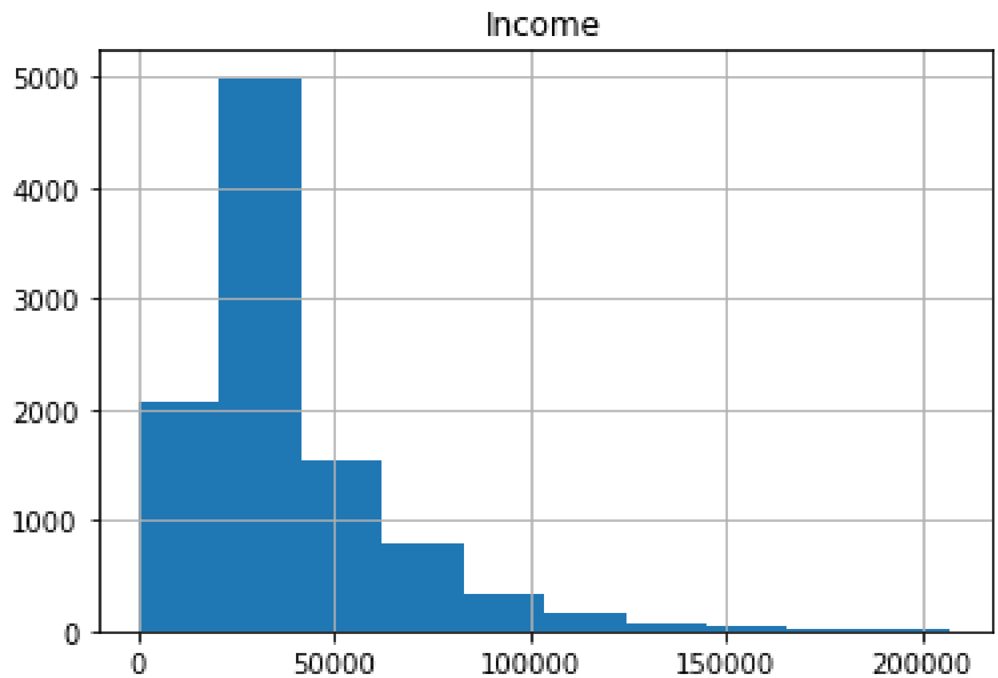

Overweight
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

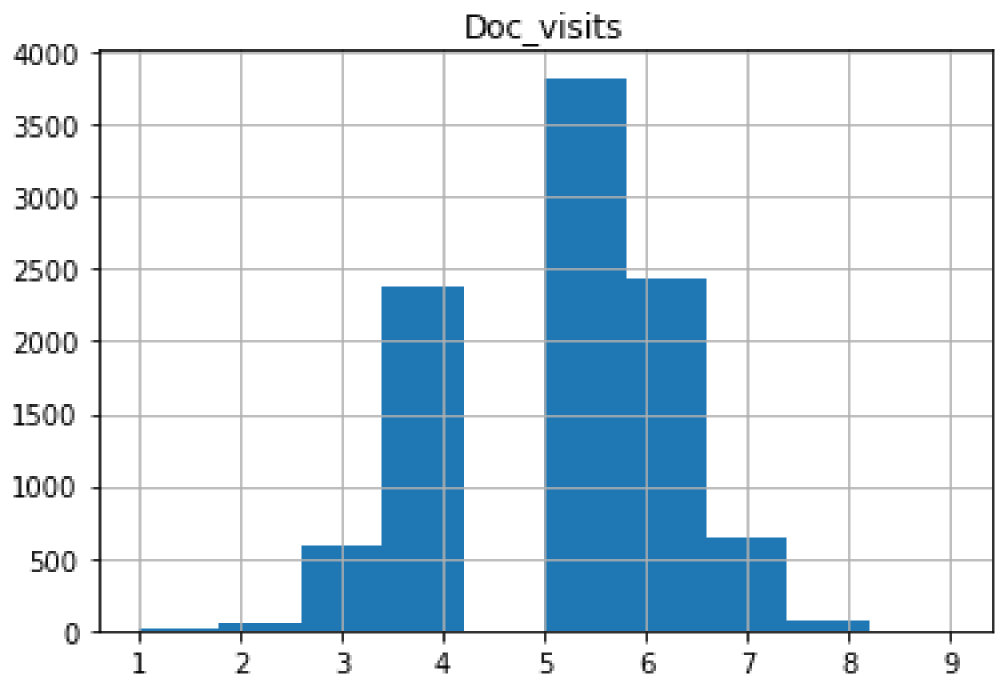

HighBlood
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

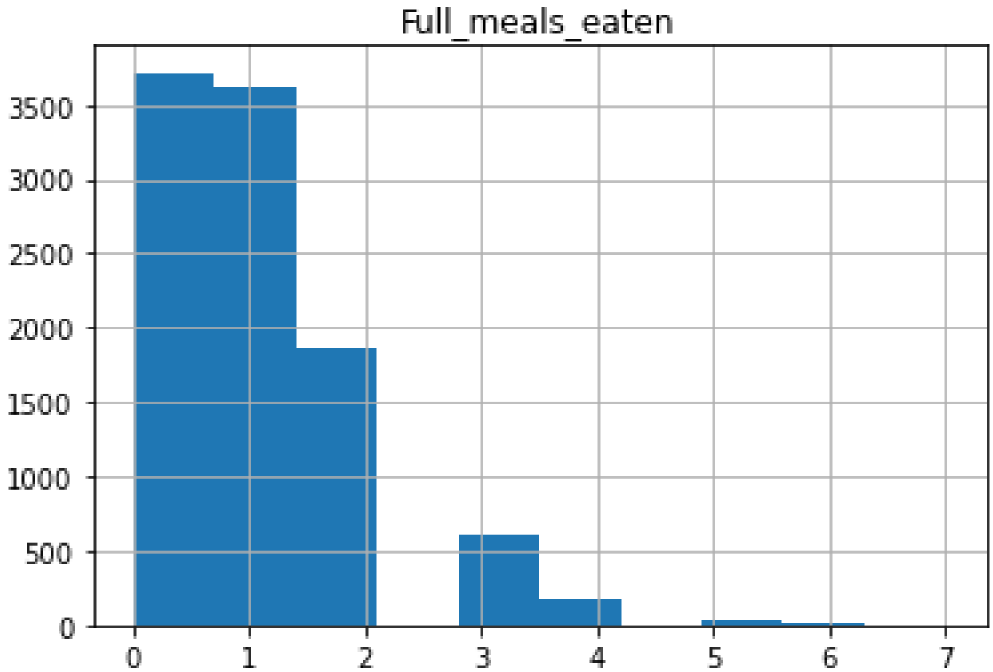

Diabetes
7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

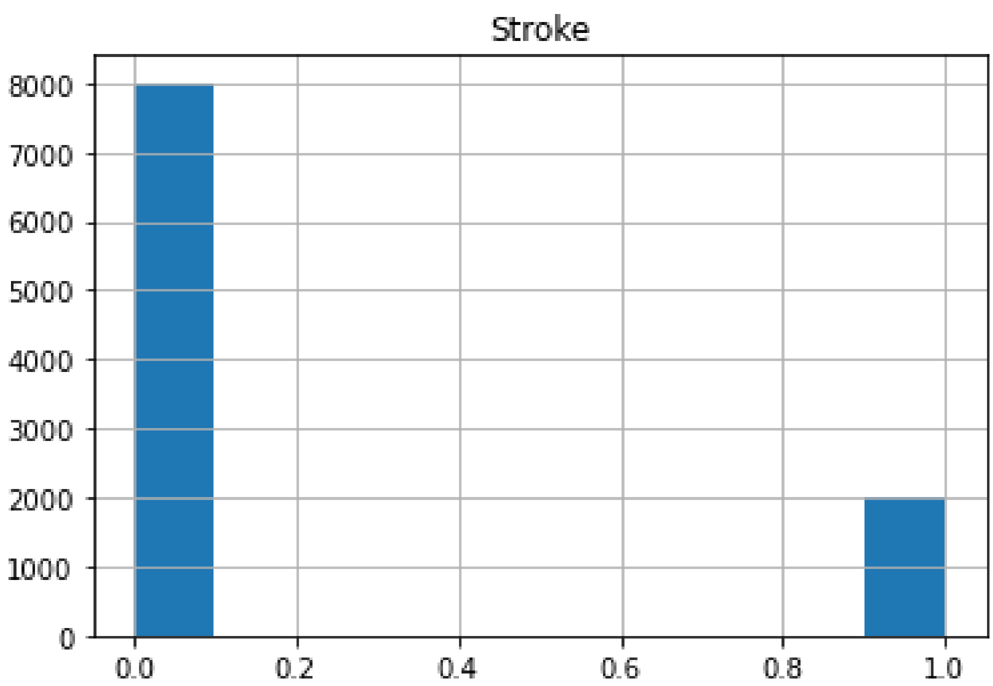

Hyperlipidemia
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

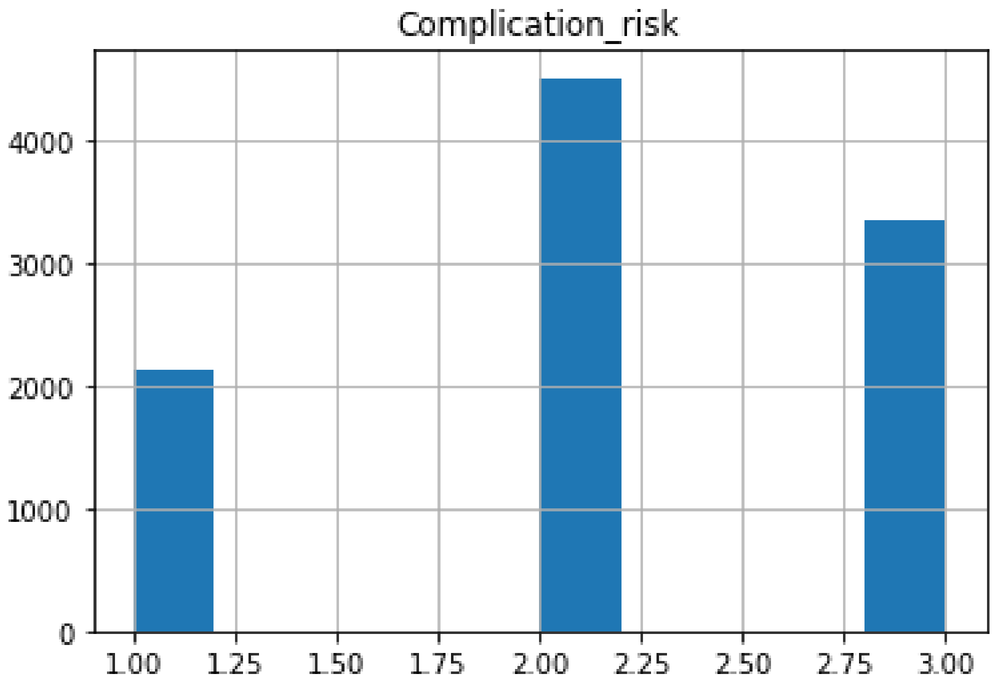

Age
3000
2500
2000
1500
1000
500
0
20
30
40
50
60
70
80
90

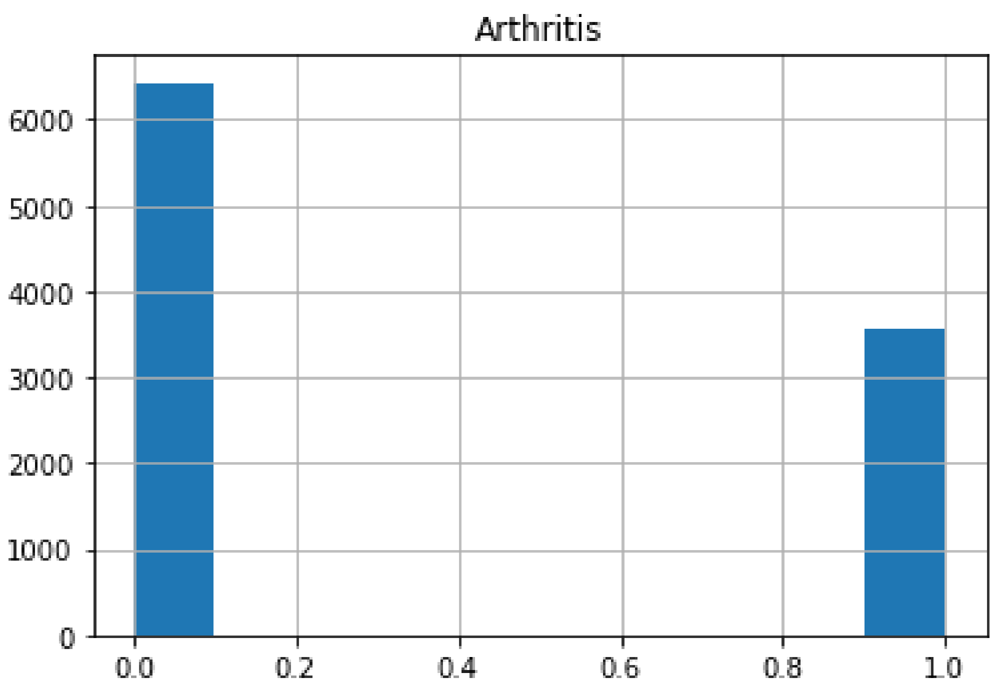

Income
5000
4000
3000
2000
1000
0
0
50000
100000
150000
200000

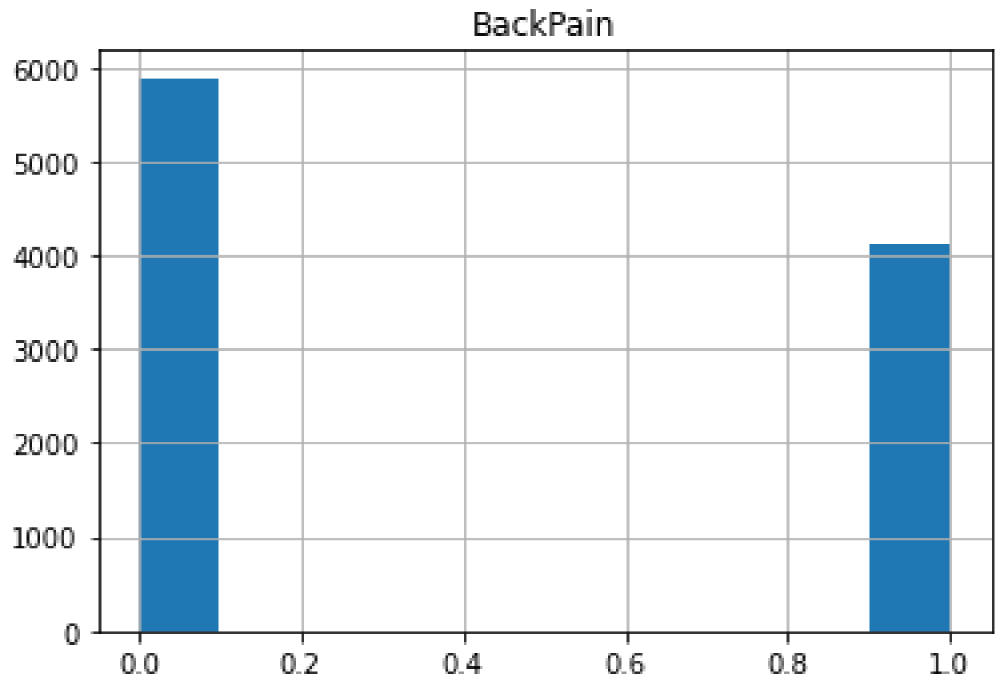

Doc_visits
4000
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9

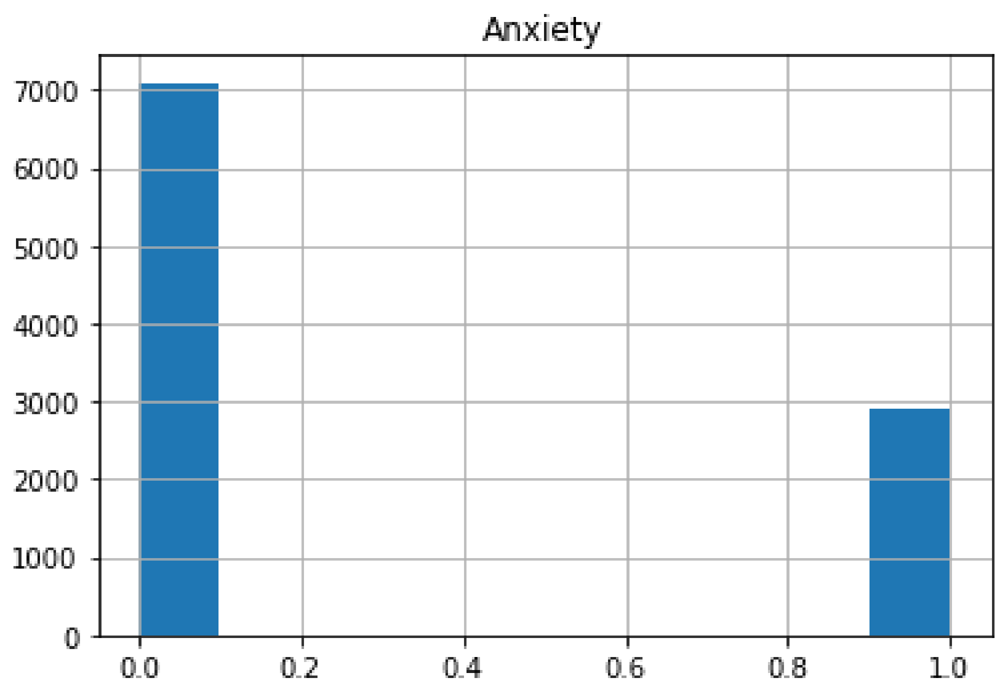

Full_meals_eaten
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7

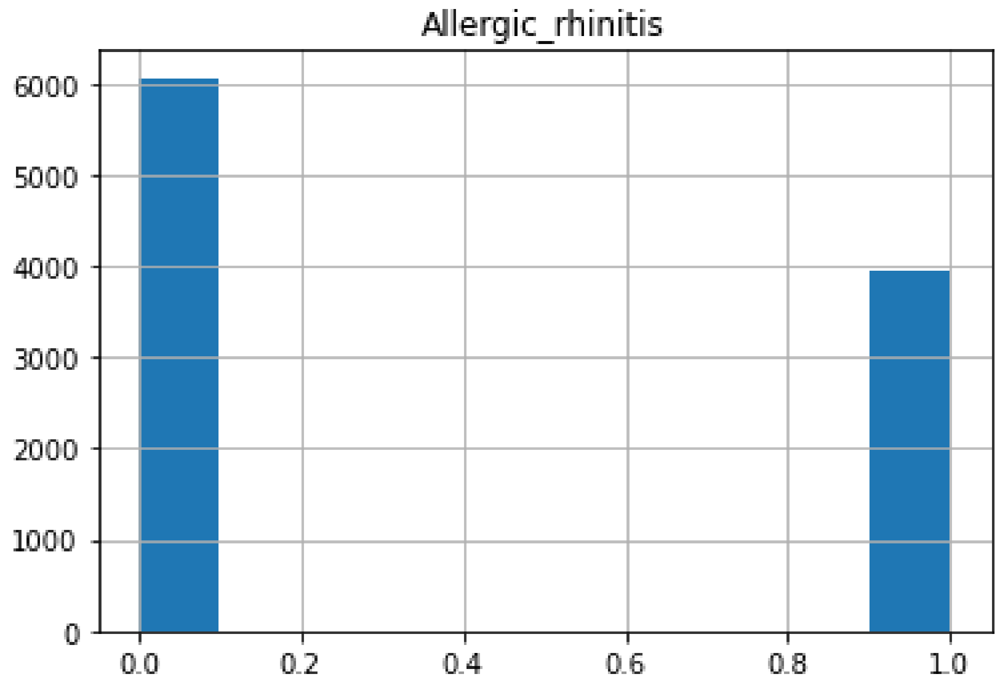

Stroke
8000
7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

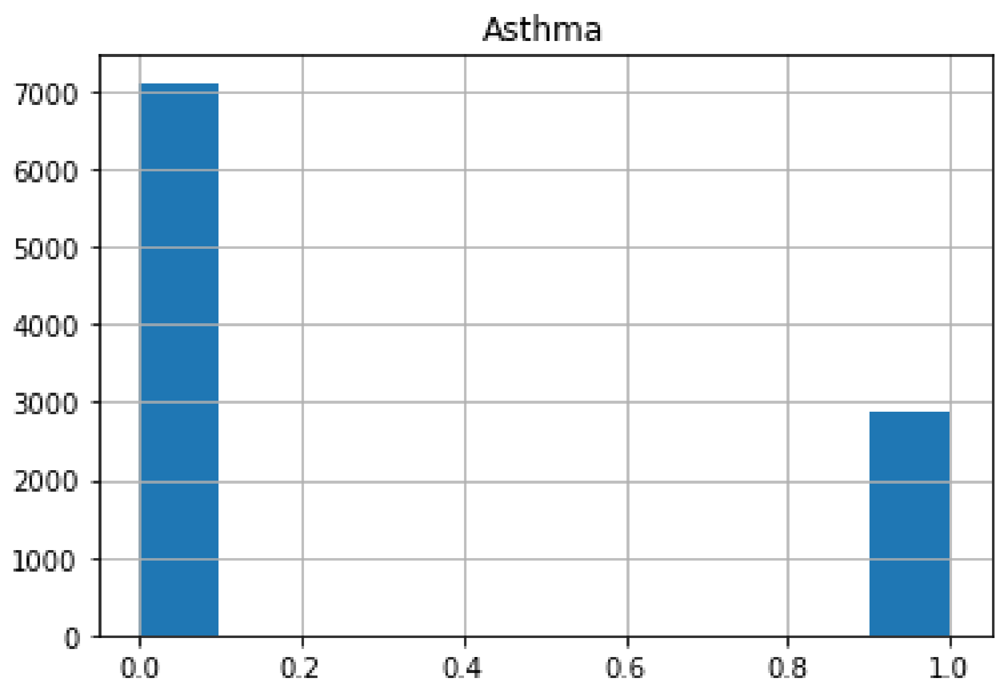

Complication_risk
4000
3000
2000
1000
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

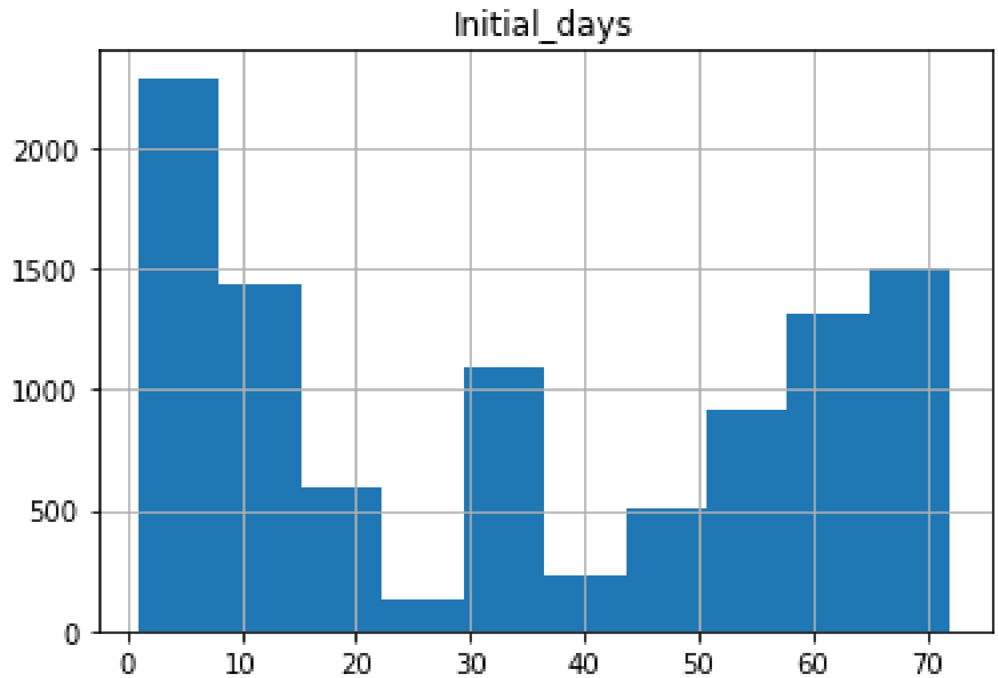

Arthritis
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

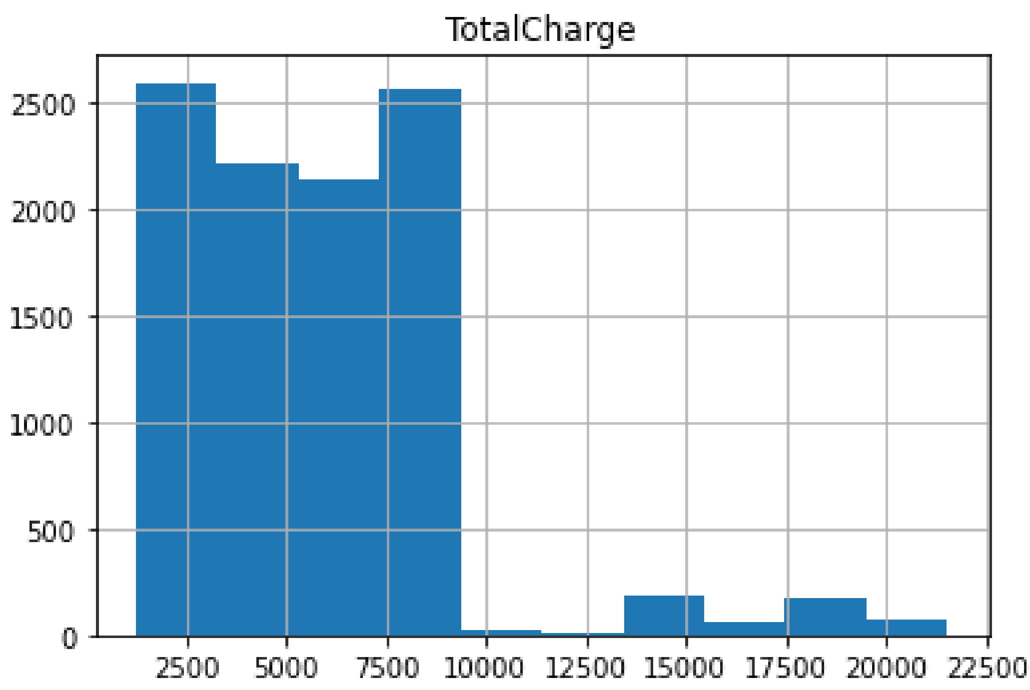

BackPain
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0


Anxiety
7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0


Allergic_rhinitis
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0


Asthma
7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0

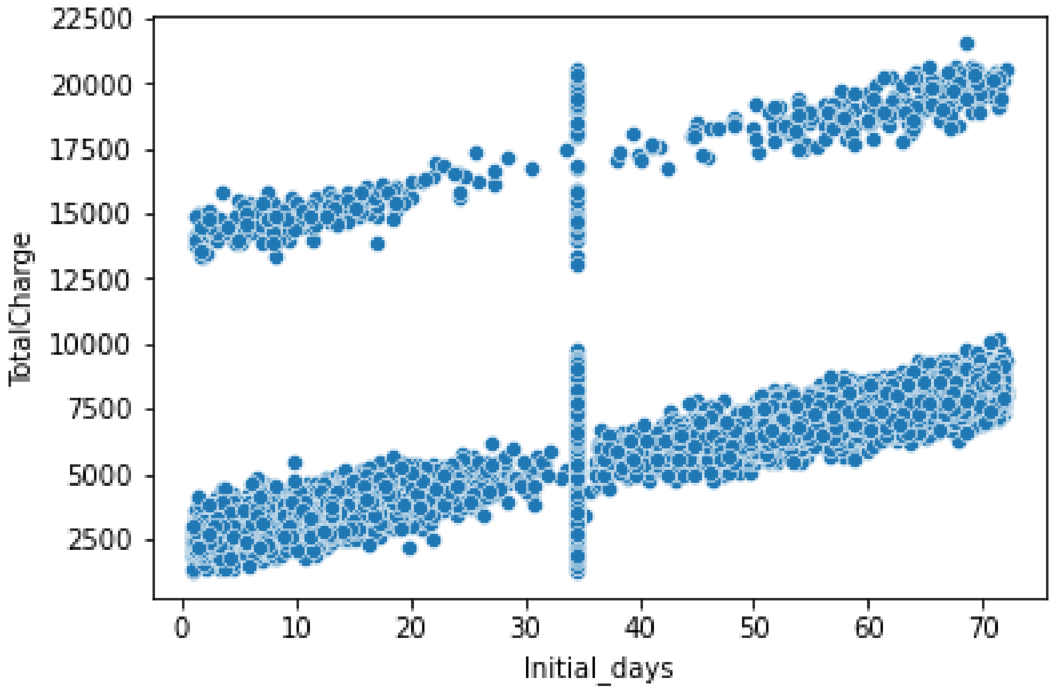

Initial_days
0
500
1000
1500
2000
0
10
20
30
40
50
60
70


TotalCharge
0
500
1000
1500
2000
2500
2500
5000
7500
10000
12500
15000
17500
20000
22500

6.3 Bivariate Statistics

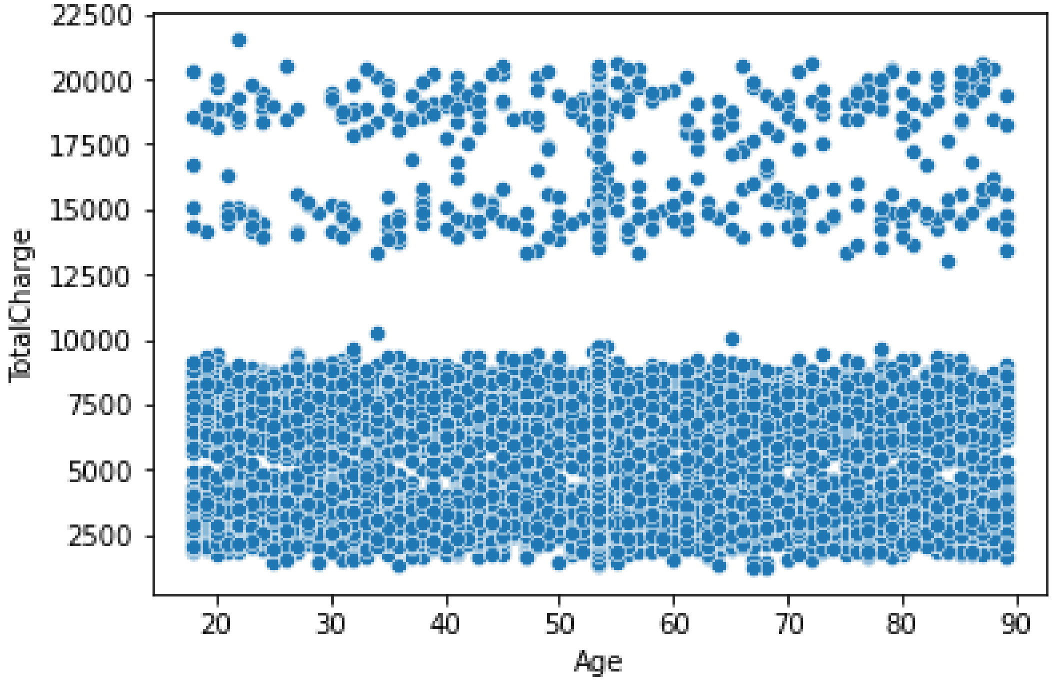
[20]:
```
# Initial days vs total charge
sns.scatterplot(x = 'Initial_days',
                y = 'TotalCharge',
                data = dmd
               )
```

[20]: <AxesSubplot:xlabel='Initial_days', ylabel='TotalCharge'>



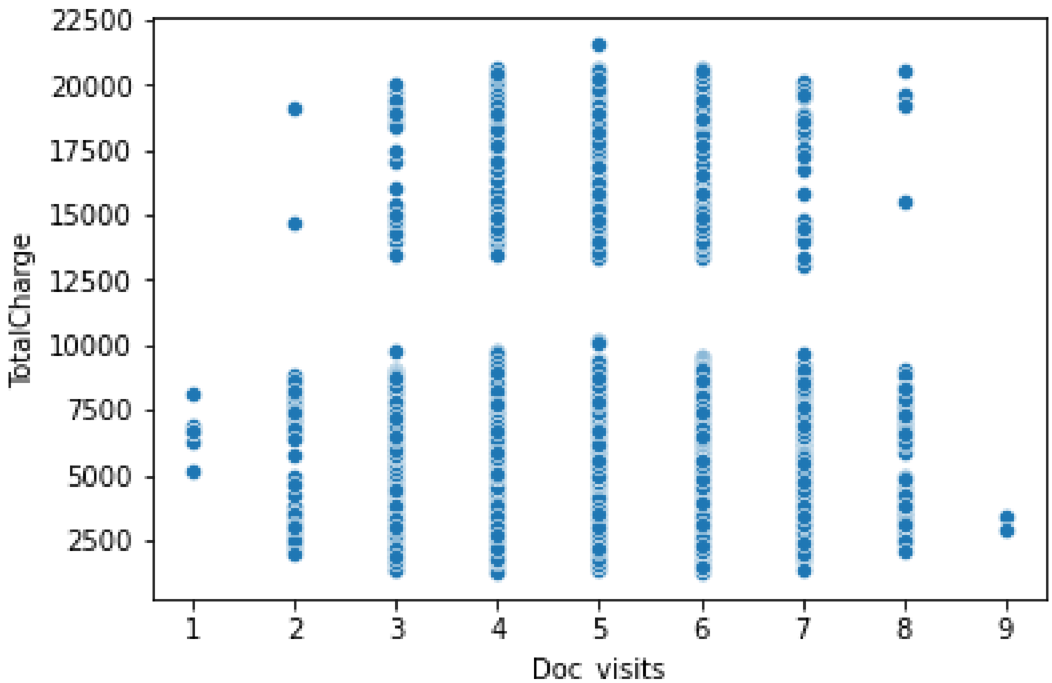
[21]:
```
# Age vs Total Charge
sns.scatterplot(x = 'Age',
                y = 'TotalCharge',
                data = dmd
               )
```

[21]: <AxesSubplot:xlabel='Age', ylabel='TotalCharge'>



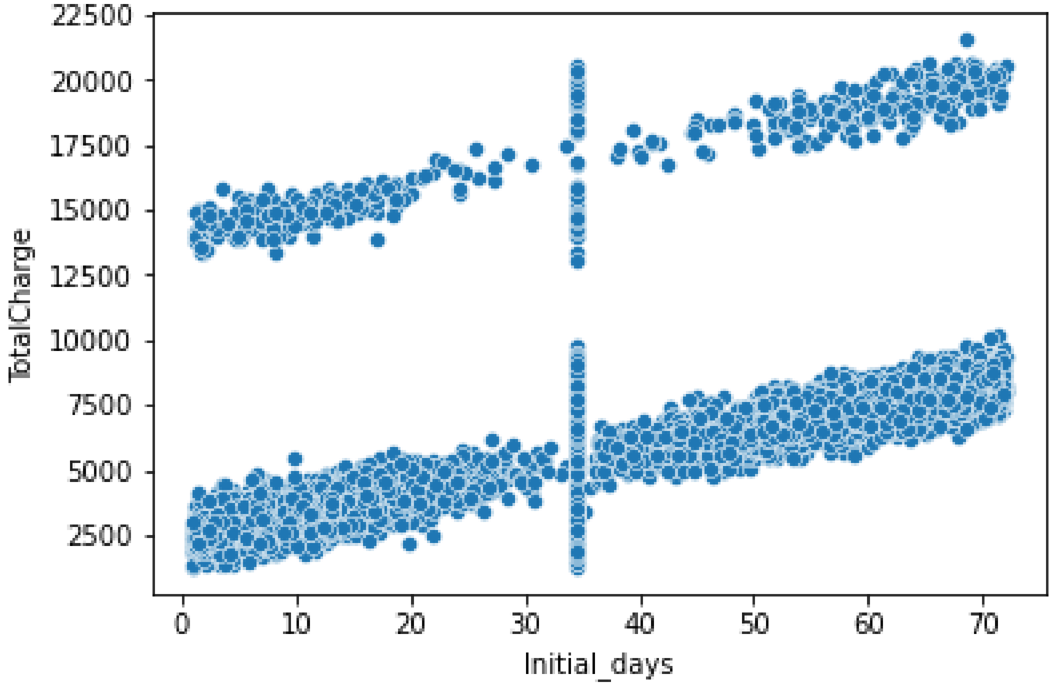
[22]:
```
# Doctor visits vs Total Charge
sns.scatterplot(x = 'Doc_visits',
                y = 'TotalCharge',
                data = dmd
               )
```

[22]: <AxesSubplot:xlabel='Doc_visits', ylabel='TotalCharge'>



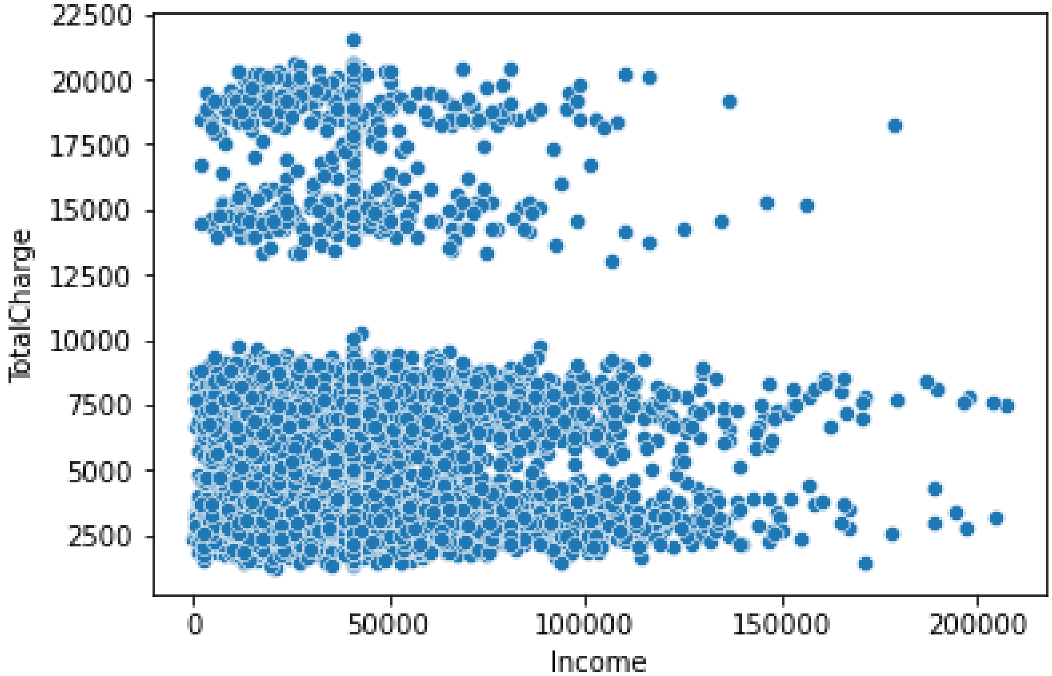
[23]:
```
# Total charges vs Total Charge
sns.scatterplot(x =  'Initial_days',
                y ='TotalCharge',
                data = dmd
               )
```

[23]: <AxesSubplot:xlabel='Initial_days', ylabel='TotalCharge'>



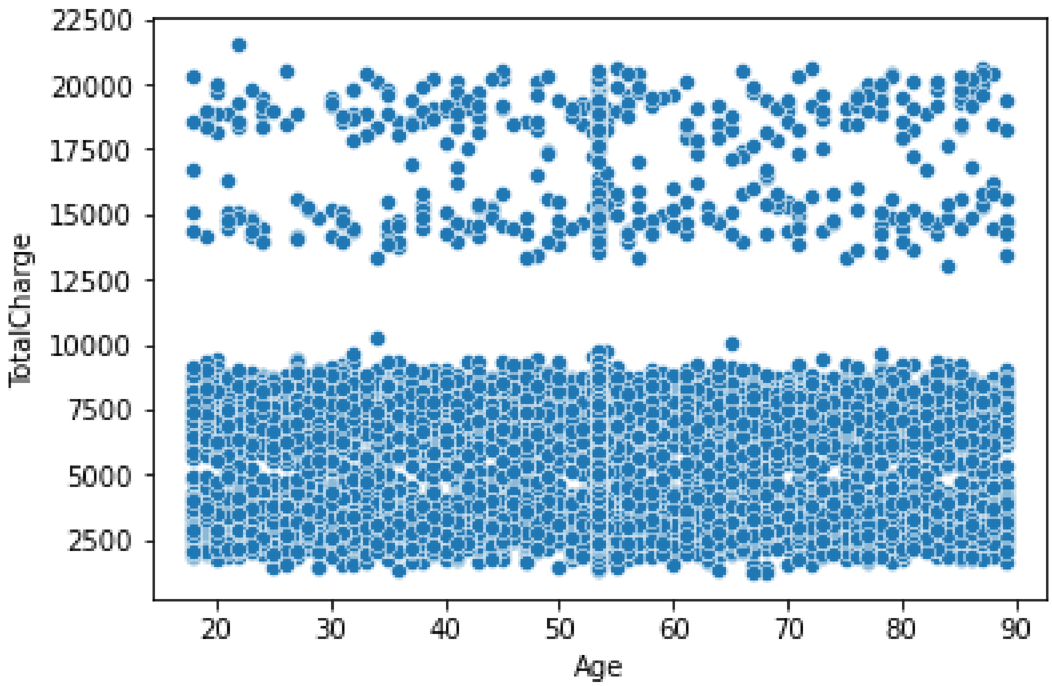
[24]:
```
# Overweight vs Total Charge
sns.scatterplot(x = 'Income',
                y = 'TotalCharge',
                data = dmd
               )
```

[24]: <AxesSubplot:xlabel='Income', ylabel='TotalCharge'>



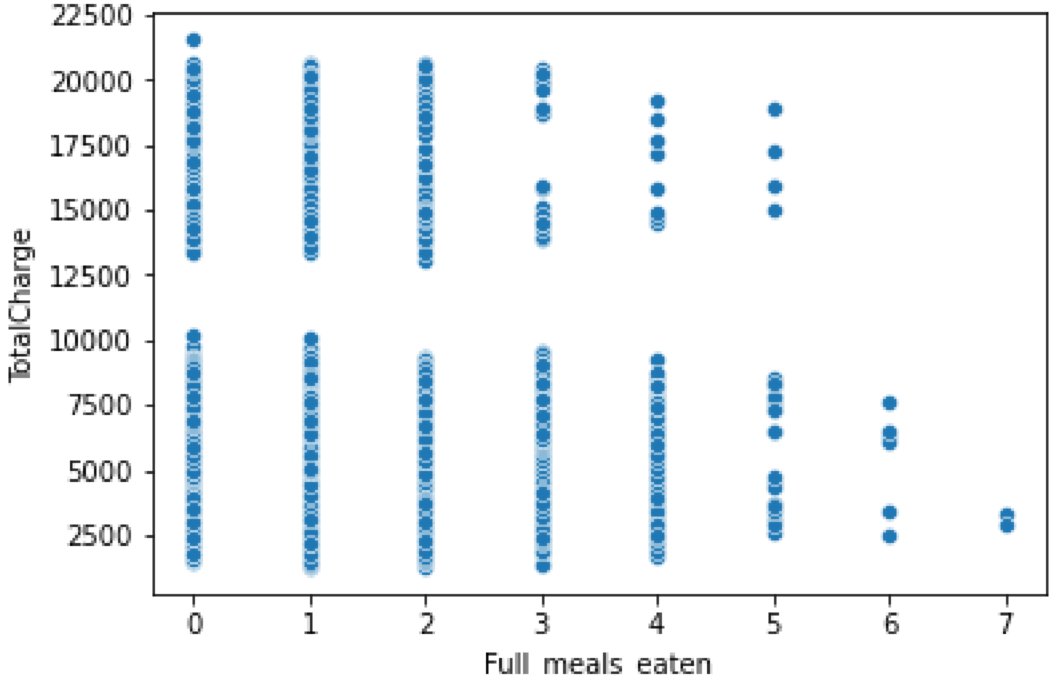
[25]:
```python
# Hyperlipidemia vs Total Charge
sns.scatterplot(x = 'Age',
                y = 'TotalCharge',
                data = dmd
               )
```

[25]: <AxesSubplot:xlabel='Age', ylabel='TotalCharge'>



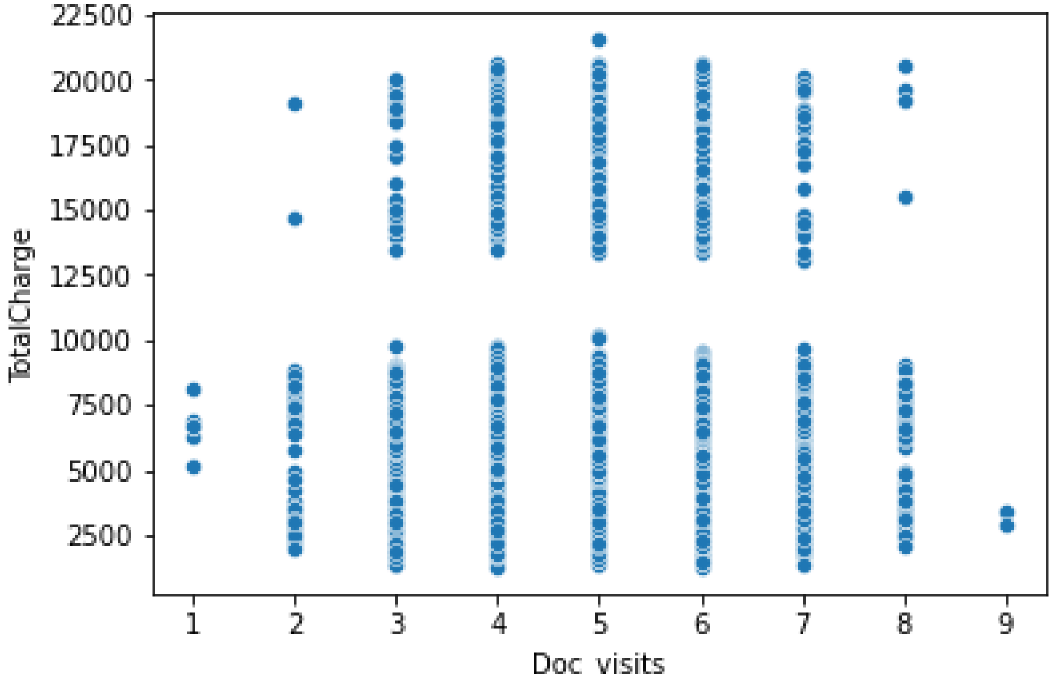
[26]:
```
# Diabetes vsTotal Charge
sns.scatterplot(x = 'Full_meals_eaten',
                y = 'TotalCharge',
                data = dmd
               )
```

[26]: <AxesSubplot:xlabel='Full_meals_eaten', ylabel='TotalCharge'>



[27]:
```
# High Blood Pressure vs Total Charge
sns.scatterplot(x = 'Doc_visits',
                y = 'TotalCharge',
                data = dmd
               )
```

[27]: <AxesSubplot:xlabel='Doc_visits', ylabel='TotalCharge'>



6.4 Save Data File

[28]:
```
dmd.to_csv('clean_data_208.csv')
```

7 Part IV: Model Comparison and Analysis

7.1 D1: Initial Model Code

[29]:
```
# Import OlS
from statsmodels.formula.api import ols
```

[30]:
```
# Create our initial model and fit a line
model = ols('TotalCharge ~ Initial_days + Income + VitD_supp + Hyperlipidemia +␣
 ↪Age + Overweight + HighBlood + Diabetes  + BackPain + Asthma + Stroke +␣
 ↪Arthritis + Reflux_esophagitis + Complication_risk + Full_meals_eaten +␣
 ↪Anxiety', dmd).fit()
model.params
```

[30]:
```
Intercept             2194.692299
Initial_days            81.991691
Income                  -0.000325
VitD_supp               35.989404
Hyperlipidemia         106.606090
Age                      2.695071
```

```
Overweight              -37.108550
HighBlood               158.137647
Diabetes                -23.418812
BackPain                 93.422118
Asthma                   76.993745
Stroke                 -102.492442
Arthritis                75.883999
Reflux_esophagitis       36.249863
Complication_risk       240.473977
Full_meals_eaten         10.727052
Anxiety                 188.010526
dtype: float64
```

[31]:

```python
# Provide output of model for analysis
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.369
Model:                            OLS   Adj. R-squared:                  0.368
Method:                 Least Squares   F-statistic:                     364.4
Date:                Mon, 03 Jan 2022   Prob (F-statistic):               0.00
Time:                        11:56:43   Log-Likelihood:                -93138.
No. Observations:               10000   AIC:                         1.863e+05
Df Residuals:                    9983   BIC:                         1.864e+05
Df Model:                          16
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept           2194.6923    147.932     14.836      0.000    1904.716
2484.669
Initial_days          81.9917      1.082     75.783      0.000      79.871
84.112
Income                -0.0003      0.001     -0.301      0.764      -0.002
0.002
VitD_supp             35.9894     42.785      0.841      0.400     -47.877
119.856
Hyperlipidemia       106.6061     56.836      1.876      0.061      -4.804
218.017
Age                    2.6951      1.494      1.804      0.071      -0.233
5.623
Overweight           -37.1086     55.974     -0.663      0.507    -146.830
72.613
HighBlood            158.1376     54.679      2.892      0.004      50.956
```

```
265.319
Diabetes             -23.4188     60.270     -0.389      0.698    -141.561
94.723
BackPain              93.4221     54.638      1.710      0.087     -13.679
200.523
Asthma                76.9937     59.294      1.299      0.194     -39.234
193.222
Stroke              -102.4924     67.266     -1.524      0.128    -234.347
29.362
Arthritis             75.8840     56.109      1.352      0.176     -34.101
185.869
Reflux_esophagitis    36.2499     54.575      0.664      0.507     -70.729
143.229
Complication_risk    240.4740     36.814      6.532      0.000     168.311
312.637
Full_meals_eaten      10.7271     26.670      0.402      0.688     -41.551
63.005
Anxiety              188.0105     59.176      3.177      0.001      72.014
304.007
==============================================================================
Omnibus:                     7177.288   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            82645.776
Skew:                           3.518   Prob(JB):                         0.00
Kurtosis:                      15.200   Cond. No.                     2.69e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.69e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

7.2 D2: Provide a Statistically Based Variable Selection Model

In order to create a reduced model, we will include any variable with a p-value less than .1. This value was selected as it is a common cut off value for statistical significance. We will recreate a model utilizing only initial days, age, hyperlipidemia status, high blood pressure status, high blood pressure, back pain, anxiety and original complication risk. The initial model has a R-squared value of 0.369.

8 Reduced Model and Residual Plot

[32]:
```
ReducedModel = ols('TotalCharge ~ Initial_days + Age + Hyperlipidemia +␣
 ↪HighBlood + Complication_risk + BackPain + Anxiety' , dmd).fit()
ReducedModel.params
```

[32]:
```
Intercept            2219.212872
Initial_days           82.026198
Age                     2.704697
Hyperlipidemia        107.292748
HighBlood             158.271848
Complication_risk     239.883611
BackPain               93.063504
Anxiety               190.297650
dtype: float64
```

[33]:
```
print(ReducedModel.summary())
```

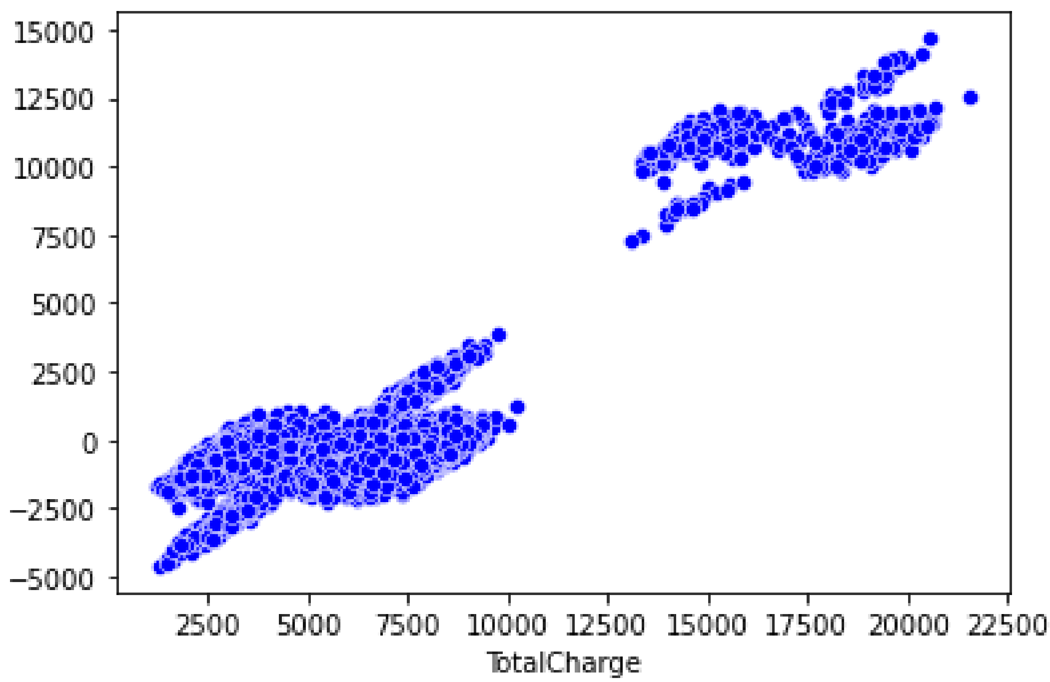
```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.368
Model:                            OLS   Adj. R-squared:                  0.368
Method:                 Least Squares   F-statistic:                     831.9
Date:                Mon, 03 Jan 2022   Prob (F-statistic):               0.00
Time:                        11:56:44   Log-Likelihood:                -93142.
No. Observations:               10000   AIC:                         1.863e+05
Df Residuals:                    9992   BIC:                         1.864e+05
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept          2219.2129    127.177     17.450      0.000    1969.919
2468.506
Initial_days         82.0262      1.081     75.899      0.000      79.908
84.145
Age                   2.7047      1.493      1.812      0.070      -0.222
5.631
Hyperlipidemia      107.2927     56.812      1.889      0.059      -4.069
218.655
HighBlood           158.2718     54.648      2.896      0.004      51.151
265.393
Complication_risk   239.8836     36.799      6.519      0.000     167.751
312.017
BackPain             93.0635     54.598      1.705      0.088     -13.960
200.087
Anxiety             190.2977     59.156      3.217      0.001      74.339
306.256
==============================================================================
Omnibus:                     7177.127   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            82631.996
```

```
Skew:                           3.518   Prob(JB):                         0.00
Kurtosis:                      15.199   Cond. No.                         324.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[34]:
```
med_df = pd.read_csv('clean_data_208.csv')
med_df['intercept'] = 1
residuals = med_df['TotalCharge'] - ReducedModel.
 ↪predict(med_df[['Initial_days', 'Age', 'Hyperlipidemia',␣
 ↪'HighBlood','Complication_risk', 'BackPain', 'Anxiety', 'intercept']])
sns.scatterplot(x = med_df['TotalCharge'], y = residuals, color = 'blue')
```

[34]: `<AxesSubplot:xlabel='TotalCharge'>`



9 E1: Explain Data Analyses Process

We utilized the p-value to decide which predictor variables to continue with. While our R-squared value is the same, we have reduced the number of variables from 16 to 7. This allows us to have a similar level of explanation of variation while reducing the possibility of the other predictors influencing the output. Both models have a much lower R-squared value than we would want to begin utilizing the model. This indicates that we need further investigation of possible predictor variables.

Looking at the residual plot that we created we can note a few things. We see indications of normal distribution and heteroscedasticity from charges of 0 to around 12,500 dollars. Then a second distribution above 12,500 with much higher residuals, indicating larger errors once charges exceed this amount.

10 F1: Regression Equation

Total Charge = 2219.21 + (82.02 * initial days) + (2.70 * age) + (107.29 * Hyperlipidemia) + (158.27 * high blood pressure) + (239.88 * complication risk) + (93.06 * back pain) + (190.30 * Anxiety)

Looking at our equation, we can observe the most impactful coefficients. The two most impactful coefficients appear to be the number of days of the initial admission and the complication risk at admission. While there are higher coefficients, we know the hyperlipidemia, back pain, high blood pressure, and anxiety are all categorical data types, meaning if the patient has them the coefficient is multiplied by one or by zero if not present. All coefficients will have a positive impact on the total charge to the patient.

10.1 Limitations of Analyses

While this model can explain some of the variance within the initial charges to a patient, it is clear from the low r-squared and multiple groupings within the residual plot, that we are missing some information. We would need to collect more data to reevaluate the model and increase its efficacy.

11 Recommendations

It is our recommendation that we continue to collect data, and to expand the data collection we are currently doing. I would recommend that you consult subject matter experts on other possible predictor variables. We currently do not have a model that I would be comfortable utilising to adress the business problem of predicting the total cost of initial admission for patients.

12 Annotations

Chantal D. Larose, & Daniel T. Larose. (2019). Data Science Using Python and R. Wiley.

Pandas Development Team. (2008). pandas.DataFrame.drop — pandas 1.3.0 documentation. Pandas. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.

Python - seaborn.residplot() method. GeeksforGeeks. (2020, August 17). Retrieved December 26, 2021, from https://www.geeksforgeeks.org/python-seaborn-residplot-method/

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Shin, T. (2021, December 4). Understanding multicollinearity and how to detect it in Python. Medium. Retrieved December 18, 2021, from https://towardsdatascience.com/everything-you-need-to-know-about-multicollinearity-2f21f082d6dc