d208task1

March 12, 2024

1 D208 Predictive Modeling Performance Assessment, Task # 1

Submitted by William J Townsend, Student ID 003397146, for WGU's MSDA program

1.1 Table of Contents

A1: Research Question

The research question that I want to examine is, "What factors most significantly contribute to the length of a hospital stay?" While this is a very broad question, it is an important question because identifying the factors which most significantly contribute to longer hospital stays allows for hospitals to better manage patients and their individualized risk factors in order to try to address these factors with the aim of reducing the lengths of hospitalization. This benefits the hospital, saving money and resources that can be allocated elsewhere, and it also benefits the patient by helping get them well more quickly.

A2: Objectives and Goals of Analysis

The objective of this analysis is to use a multiple regression model to determine which factors in the dataset (the independent or explanatory variables) contribute most significantly to the length of hospitalization (the dependent or target variable). Identification of these factors can inform the way the hospital handles certain patients to be more responsive to their individual risk factors, in order to generate better outcomes for everyone involved. Identification of larger contributory factors can lead to allocation of resources towards these factors, helping patients recover faster and reducing hospital stays, which frees up resources that would've been spent on longer hospitalizations. If length of hospitalization is a primary contributor to readmittance rates, which makes intuitive sense, then identification of these factors may also reduce financial penalties imposed by the state upon the hospital, by reducing readmission as a result.

B1: Assumptions of a Multiple Regression Model

Simple linear regression is a good starting point for understanding multiple regression. If you know that driving 50 miles takes you 1 hour, and driving 100 miles takes 2 hours, we can start to form a relationship that lets us predict that driving 75 miles take us 1.5 hours. This involves a single explanatory variable (distance travelled) and a single target variable (time taken), and anyone who has driven very much recognizes that its also overly simplistic. Other variables might be involved which could impact that target variable, such as traffic density, construction zones, the speed at which you're driving, whether you're driving on a highway or in a city, the weather, and many more other variables all contribute to how long it takes to drive somewhere. Multiple regression goes beyond simple linear regression, aiming to try to handle all of these different variables to ascertain how each contributes (the relationship) to the overall output (time taken) so that we can more accurately predict and model how long it would take us to travel a given distance.

Creating a multiple regression model is a process which aims to simplify several different variables and their relationships with a target variable. In order to do this, multiple regression models make several assumptions of the data that is fed into them. If our data does not actually meet these assumptions, then we end up with a Garbage In, Garbage Out situation where our model will provide misleading results. These assumptions are (Statology, 2021):

- *Linear relationships must exist between each explanatory variable (x) and the response variable (y)* A plot of the x and y variables must demonstrate a linear relationship. If no relationship exists between x1 and y, then there's no relationship for the multiple regression to tease out for x1 and y when compared with x2, x3, and so on. If the color of a car has no relationship with the time taken to drive a distance, then there's nothing to be gained by including it in a multiple regression analysis.
- *None of the explanatory (x) variables being compared can be significantly correlated with each other (multicolinearity)* Where x variables are closely related to each other, this confounds the multiple regression model's attempt to find a relationship between x and y, because

the different x variables are essentially "talking" to each other throughout the process. If construction zones encountered are highly correlated with traffic density because they reduce available lanes in an area, then one of the two variables would need to be removed in a multiple regression model.

- *The observations are independent of each other* Each row in a data set represents an observation. If a driver takes 37 minutes to drive 70 miles, encountering a certain traffic density, at certain speeds, in specific weather, this cannot have an impact (must be independent) on the next observation of a driver going on another trip, in another (but possibly the same) weather, etc. If an observation is related to a prior observation, we run into a similar problem as multicolinearity, where different target y variables are "talking" to each other throughout the process of trying to find a relationship between x and y.
- *The model's residuals (variance from the line of best fit) have a normal distribution* When considering a line of best fit, there is usually some amount of variance or error along the vertical axis between the data points and the line of best fit. If you were to measure this variance for every single data point, this would create a new data set, and a plot of that distribution needs to be a normal distribution, in order to use multiple regression.
- *At every point in the linear model, residuals have a constant variance (homoscedasticity)* Homoscedasticity refers to the tendency for a model's residuals (the variance along the vertical axis from the line of best fit) to be relatively constant. If these residuals are not constant and instead vary significantly, then the residuals are actually heteroscedastic. If heteroscedasticity undermines the multiple regression model, making it unreliable.

The first three of these assumptions are relatively straightforward - it is hard to measure a relationship where no relationship exists (the first assumption) or where the relationship is changing while you're trying to measure it (the second and third assumptions). The last two assumptions, regarding normal distributions of residuals and homoscedasticity are much more complex, and I initially struggled with the idea of how the two would work - if the residuals need to be constant, how would we end up with a normal distribution?

The key is that homoscedasticity doesn't necessarily require the residuals from a model to be *literally* constant, but *relatively* constant. Mostly, this ends up meaning that there cannot be significant outliers within the data. I understood this by imagining a model where the residuals were homoscedastic and normally distributed around a mean error of 10 units. This means that most residuals would be at an error of 10 units, many would be at an error of 9 or 11 units, some would be at 8 or 12 units, and very few would be at 7 or 13 units. This is obviously a normal distribution. It is also homoscedastic becuase it is also nearly constant at about 10 units of error, because there are not a cluster of errors at 20 units (or even more extreme), which would represent an outlier in the dataset and ruin the normal distribution.

B2: Benefits of Chosen Tools

I will be using Python throughout this analysis project. Python is a programming language that supports data science processes very well, particularly in the use of packages designed specifically for this. Python also happens to be the only programming language that I know to any sort of significant degree. I'll also be using several Python packages to perform this analysis: - pandas allows for the handling of the dataset in something like a large table or spreadsheet - NumPy allows for performing certain mathematical operations or assignment of certain values within the dataset - Seaborn and MatPlotLib provide graphing functionality - SciPy's statsmodels provides several important functionalities for both the multiple regression model such as graphing residuals, as well as for checking problems such as multicolinearity using the variance inflation factor - sklearn's preprocessing is useful for transforming our data, when needed

[ ]:
```
import pandas as pd
from pandas.api.types import CategoricalDtype
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn import preprocessing

# The CSV's first column is an index and Pandas will duplicate this and create␣
↪an column without 'index_col=0'
df = pd.read_csv('./medical_clean.csv', index_col=0)
# Check data types and number of values, as well as overall size of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 10000
Data columns (total 49 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Customer_id         10000 non-null  object
 1   Interaction         10000 non-null  object
 2   UID                 10000 non-null  object
 3   City                10000 non-null  object
 4   State               10000 non-null  object
 5   County              10000 non-null  object
 6   Zip                 10000 non-null  int64
 7   Lat                 10000 non-null  float64
 8   Lng                 10000 non-null  float64
 9   Population          10000 non-null  int64
 10  Area                10000 non-null  object
 11  TimeZone            10000 non-null  object
 12  Job                 10000 non-null  object
 13  Children            10000 non-null  int64
 14  Age                 10000 non-null  int64
 15  Income              10000 non-null  float64
 16  Marital             10000 non-null  object
 17  Gender              10000 non-null  object
 18  ReAdmis             10000 non-null  object
 19  VitD_levels         10000 non-null  float64
 20  Doc_visits          10000 non-null  int64
 21  Full_meals_eaten    10000 non-null  int64
 22  vitD_supp           10000 non-null  int64
```

```
 23  Soft_drink          10000 non-null  object
 24  Initial_admin       10000 non-null  object
 25  HighBlood           10000 non-null  object
 26  Stroke              10000 non-null  object
 27  Complication_risk   10000 non-null  object
 28  Overweight          10000 non-null  object
 29  Arthritis           10000 non-null  object
 30  Diabetes            10000 non-null  object
 31  Hyperlipidemia      10000 non-null  object
 32  BackPain            10000 non-null  object
 33  Anxiety             10000 non-null  object
 34  Allergic_rhinitis   10000 non-null  object
 35  Reflux_esophagitis  10000 non-null  object
 36  Asthma              10000 non-null  object
 37  Services            10000 non-null  object
 38  Initial_days        10000 non-null  float64
 39  TotalCharge         10000 non-null  float64
 40  Additional_charges  10000 non-null  float64
 41  Item1               10000 non-null  int64
 42  Item2               10000 non-null  int64
 43  Item3               10000 non-null  int64
 44  Item4               10000 non-null  int64
 45  Item5               10000 non-null  int64
 46  Item6               10000 non-null  int64
 47  Item7               10000 non-null  int64
 48  Item8               10000 non-null  int64
dtypes: float64(7), int64(15), object(27)
memory usage: 3.8+ MB
```

[ ]:
```python
# Visually inspect dataframe to facilitate exploration, spot problems
pd.set_option("display.max_columns", None)
df.head(5)
```

[ ]:
```
          Customer_id                           Interaction  \
CaseOrder
1             C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
2             Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
3             F995323  a2057123-abf5-4a2c-abad-8ffe33512562
4             A879973  1dec528d-eb34-4079-adce-0d7a40e82205
5             C544523  5885f56b-d6da-43a3-8760-83583af94266

                                        UID         City State        County  \
CaseOrder
1          3a83ddb66e2ae73798bdf1d705dc0932          Eva    AL        Morgan
2          176354c5eef714957d486009feabf195     Marianna    FL       Jackson
3          e19a0fa00aeda885b8a436757e889bc9  Sioux Falls    SD     Minnehaha
4          cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca
```

```
5          d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William

             Zip       Lat       Lng  Population      Area         TimeZone  \
CaseOrder
1          35621  34.34960 -86.72508        2951  Suburban   America/Chicago
2          32446  30.84513 -85.22907       11303     Urban   America/Chicago
3          57110  43.54321 -96.63772       17125  Suburban   America/Chicago
4          56072  43.89744 -93.51479        2162  Suburban   America/Chicago
5          23181  37.59894 -76.88958        5287     Rural  America/New_York

                                        Job  Children  Age    Income  \
CaseOrder
1          Psychologist, sport and exercise         1   53  86575.93
2              Community development worker         3   51  46805.99
3                   Chief Executive Officer         3   53  14370.14
4                       Early years teacher         0   78  39741.49
5             Health promotion specialist         1   22   1209.56

            Marital  Gender ReAdmis  VitD_levels  Doc_visits  \
CaseOrder
1          Divorced    Male      No    19.141466           6
2           Married  Female      No    18.940352           4
3           Widowed  Female      No    18.057507           4
4           Married    Male      No    16.576858           4
5           Widowed  Female      No    17.439069           5

           Full_meals_eaten  vitD_supp Soft_drink        Initial_admin  \
CaseOrder
1                         0          0         No  Emergency Admission
2                         2          1         No  Emergency Admission
3                         1          0         No   Elective Admission
4                         1          0         No   Elective Admission
5                         0          2        Yes   Elective Admission

          HighBlood Stroke Complication_risk Overweight Arthritis Diabetes  \
CaseOrder
1               Yes     No            Medium         No       Yes      Yes
2               Yes     No              High        Yes        No       No
3               Yes     No            Medium        Yes        No      Yes
4                No    Yes            Medium         No       Yes       No
5                No     No               Low         No        No       No

          Hyperlipidemia BackPain Anxiety Allergic_rhinitis  \
CaseOrder
1                     No      Yes     Yes               Yes
2                     No       No      No                No
3                     No       No      No                No
```

```
4                       No       No      No                No
5                      Yes       No      No               Yes

          Reflux_esophagitis Asthma     Services  Initial_days  TotalCharge  \
CaseOrder
1                         No    Yes   Blood Work     10.585770  3726.702860
2                        Yes     No  Intravenous     15.129562  4193.190458
3                         No     No   Blood Work      4.772177  2434.234222
4                        Yes    Yes   Blood Work      1.714879  2127.830423
5                         No     No      CT Scan      1.254807  2113.073274

           Additional_charges  Item1  Item2  Item3  Item4  Item5  Item6  \
CaseOrder
1                17939.403420      3      3      2      2      4      3
2                17612.998120      3      4      3      4      4      4
3                17505.192460      2      4      4      4      3      4
4                12993.437350      3      5      5      3      4      5
5                 3716.525786      2      1      3      3      5      3

           Item7  Item8
CaseOrder
1              3      4
2              3      3
3              3      3
4              5      5
5              4      3
```

B3: Justification of Multiple Regression

Multiple linear regression is used for modelling the relationship between a continuous dependent variable against other independent variables, which may be either continuous or categorical. In this case, I want to examine the relationship between several variables and the length of hospitalization, and the length of hospitalization is a continuous variable. Hospitalization has no particular finite maximum, and it can also be measured with increasing precision (from days to hours to minutes and even possibly below that). The length of hospitalization, as provided in this dataset, is a floating point number which provides us the number of whole days and a significant level of precision in the measurement of partial days. As such, it should be suitable for multiple regression, allowing it to be compared to a number of potential explanatory variables.

C1: Data Preparation Goals and Necessary Manipulation

While the data set provided by WGU is described as being "clean", it is not very well cleaned, including missing several fixes that were performed in D206: Data Cleaning. For example, zip codes are stored as an int64, eliminating leading zeros and assumed to be whole numbers, rather than the qualitative strings that they actually should be. Similarly, timezones are storedin 26 different non-descriptive "timezones" which should instead be standardized to the 9 US time zones. I will reapply much of the code that I generated in my D206 project to fix these issues, with some modifications.

Multiple regression analysis requires numerical values be provided rather than strings for the various categorical and boolean types of data in this data set. The booleans can easily be remapped from True/False values to either 1/0 numeric values. The categorical variables will have to be handled differently, based upon whether or not they are ordinal (the order of the category values matters, such as "big", "bigger", "biggest") or nominal (order does not matter) in nature.

There are several different categorical data types in this data set which are not boolean in nature (already handled). Of these columns, only the survey scores columns are ordinal in nature. The original intention for these columns as described in the data dictionary was that 1 = "most important" and "8" = "least important", which is somewhat unintuitive when they are numerically represented. As a result, what I'll do is actually remap these values to reflect a more intuitive pattern where `1 < 8` is actually True. Once that is done, then I can convert these columns to an ordered categorical datatype similar to what was done in D208.

For the nominal categorical columns, those columns which are going to be used in this analysis require the generation of dummy columns, which represent categorical data in a binary numeric form. For example, the gender column contains values for Male, Female, and Nonbinary. Two columns can be created which will actually communicate the same information. If a 1 appears in the first column, then the patient is female. If a 1 appears in the second column, then the patient is nonbinary. If a 1 appears in neither column (both 0), then the patient is male. This process, called one hot encoding (WGU Courseware Resources), allows us to easily represent string data in a numeric fashion that the multiple regression analysis can handle. This can easily be done by using pandas' `get_dummies()` function.

These are the primary changes required for performing a multiple regression analysis on this dataset. Some other functions will be run to verify that the data is ready for multiple regression analysis, such as `info()` to make sure that there are no null values and verify the datatypes for each column, `value_counts()` to check all of the values in a column, or `describe()` to display summary statistics for a numeric column.

C2: Summary Statistics

The variables which I intend to examine for this analysis include the initial_days variable (the dependent variable, y) and the following independent variables (the explanatory variables, x):

- Children This variable was examined in the D207 project. Where I expected to see a significantly decreasing curve as number of children increased, we instead saw several plateaus, with similar numbers patients with 5 - 10 children.

[ ]:
```
df.Children.value_counts().sort_index()
```

[ ]:
```
0    2548
1    2509
2    1475
3    1489
4     995
5     169
6     191
7     213
8     209
9     108
```

```
10      94
Name: Children, dtype: int64
```

- Age One noticeable thing about our data from the summary statistics is that no patients are under the age of 18. As children do get hospitalized, this means that our data excludes a portion of those people who get hospitalized. This is an important exclusion to keep in mind as it pertains to the limitations of our analysis, in that it may not be representative of this portion of the overall population because they were excluded from the sample. If I had a possible means to fix this, I would, but I am constrained by the provided dataset.

[ ]:
```
df.Age.describe()
```

[ ]:
```
count    10000.000000
mean        53.511700
std         20.638538
min         18.000000
25%         36.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

- Income Income has some oddly particular numbers, which could stand to just be rounded to the nearest whole dollar. The high and low ends of this distribution seem a bit extreme. We can take a look at the 20 largest and smallest incomes to see if this is some sort of obvious outlier, or if the data really does reflect these sorts of extremes.

[ ]:
```
df.Income.describe()
```

[ ]:
```
count     10000.000000
mean      40490.495160
std       28521.153293
min         154.080000
25%       19598.775000
50%       33768.420000
75%       54296.402500
max      207249.100000
Name: Income, dtype: float64
```

[ ]:
```
# Those min and max values seem somewhat extreme, let's take a look at the␣
↪smallest...
df.Income.nsmallest(n=20)
```

[ ]:
```
CaseOrder
822      154.08
9809     300.79
288      395.23
111      401.86
```

```
8659     493.04
9129     695.22
5894     702.16
3484     798.98
1216     826.01
6300     881.07
1035     881.40
3457     953.74
6107    1048.43
3259    1078.12
2514    1196.72
5       1209.56
9088    1277.08
2299    1286.25
1384    1301.34
9029    1366.98
Name: Income, dtype: float64
```

[ ]:
```
# ...and the largest values
df.Income.nlargest(n=20)
```

[ ]:
```
CaseOrder
8387    207249.10
842     204542.41
8599    203774.60
6407    197675.00
1779    197576.18
7493    196915.60
4332    194796.24
7245    190110.80
4408    189416.27
3074    189129.92
7716    186791.40
8530    186752.00
6600    183037.10
9588    179543.70
7405    178945.40
174     178470.63
2323    174745.85
6695    171680.20
623     171288.05
9480    171031.20
Name: Income, dtype: float64
```

With many results occurring at both the low end and high end of Income levels, it seems that those min and max values are legitimate. The low end is not actually an outlier in that it is within 2 standard deviations of the mean (within 1.5, even), though the high end could be considered outliers because they are several standard deviations above the mean. However, this is legitimate data, and it makes intuitive sense because where income is concerned, there is a floor (0) but no ceiling as it pertains to what these values can be, and within the scope of American incomes, while $210,000 annual income is definitely above the mean, there is a sizeable portion of the population with incomes at or even well above this level. This is legitimate data, and legitimate data should generally be observed and handled, thus it will remain in the dataset.

One other possibility here is to instead group these values into buckets, where anything above a certain mark (say, \$150,000) would simply be another bucket, without giving particular attention or weight to the long tail of incomes ranging upwards from the hundred thousands to the billions. In a more life-like analysis, this would probably be the preferred solution because there is little difference in distinguishing between someone who makes \$80,000 and someone who makes \$80,001. However, this would turn the variable from a quantitative one into a qualitative one, and the rubric for this project requires me to use some continuous quantitative variables, which this dataset does not have a lot of. As a result, I'm going to keep this variable the way it is, despite knowing that it could be potentially improved. Besides that, as I discussed at length in my D206 project, the best solution here is to actually drop the entire column because a patient's income should have zero bearing on the care they receive while hospitalized.

- Gender I mentioned in my D206 project that this variable is flawed in the way it chooses to break down the variety of genders to only 3 possibilities. Nonetheless, this is the data that we have, so this cannot really be helped. The data reflects that about 50% of patients are female, nearly 48% are male, and just over 2% are nonbinary.

[ ]:
```
df.Gender.value_counts()
```

[ ]:
```
Female       5018
Male         4768
Nonbinary     214
Name: Gender, dtype: int64
```

- Vitamin D Level These statistics show a mean very close to 18, with a standard deviation of just over 2. The minimum and the maximum are just outside of 4 standard deviations away from the mean, with an inner quartertile range between 16.6 and 19.4. This reflects a distribution that is approximately normal.

[ ]:
```
df.VitD_levels.describe()
```

[ ]:
```
count    10000.000000
mean        17.964262
std          2.017231
min          9.806483
25%         16.626439
50%         17.951122
75%         19.347963
max         26.394449
Name: VitD_levels, dtype: float64
```

- Doctor Visits Here, we see a range from 1 - 9, with a mean of 5. The standard deviation of ~1, with an inner quartile range of 4 - 6, also reflects an approximately normal distribution.

[ ]:
```
df.Doc_visits.describe()
```

[ ]:
```
count    10000.000000
mean         5.012200
std          1.045734
min          1.000000
25%          4.000000
50%          5.000000
75%          6.000000
max          9.000000
Name: Doc_visits, dtype: float64
```

- Reason for Initial Admission Just over half of hospitalized patients in this dataset are initially admitted for an emergency. This is consistent with what we would expect (or might even be lower than we'd expect), because the hospital is generally where we go for emergencies, which we obviously try to avoid having. The remaining half of hospitalized patients are split almost evenly between being being admitted for observation or for an elective procedure.

[ ]:
```
df.Initial_admin.value_counts().sort_index()
```

[ ]:
```
Elective Admission       2504
Emergency Admission      5060
Observation Admission    2436
Name: Initial_admin, dtype: int64
```

- Complication Risk Approximately 45% of patients have a Medium risk of complications during their hospitalization. Only 21% have a Low risk of complication, while the remaining 34% have a High risk of complication.

[ ]:
```
df.Complication_risk.value_counts().sort_index()
```

[ ]:
```
High      3358
Low       2125
Medium    4517
Name: Complication_risk, dtype: int64
```

- Arthritis 35% of hospitalized patients in this dataset have arthritis.

[ ]:
```
df.Arthritis.value_counts()
```

[ ]:
```
No     6426
Yes    3574
Name: Arthritis, dtype: int64
```

- Diabetes 27% of the patients in this dataset have been diagnosed with diabetes.

[ ]:
```
df.Diabetes.value_counts()
```

[ ]:
```
No     7262
Yes    2738
```

```
Name: Diabetes, dtype: int64
```

- Back Pain Within the dataset, 41% of patients have chronic back pain, while the other 59% do not.

[ ]:
```
df.BackPain.value_counts()
```

[ ]:
```
No     5886
Yes    4114
Name: BackPain, dtype: int64
```

- Avg Daily "Total" Charge The mean daily charge in USD to patients (excluding "additional" charges) is just over \$5,300. With a standard deviation of nearly \$2,200, the minimum of ~\$1,900 is about 1.5 standard deviations from the mean. The maximum of ~\$9,200 is about 1.8 standard deviations from the mean. This indicates that the distribution is pretty tightly concentrated around the mean, but it isn't quite normal in its distribution, skewing slightly to one side. A quick check of the largest values in the dataset shows that this maximum is not an isolated outlier.

[ ]:
```
df.TotalCharge.describe()
```

[ ]:
```
count    10000.000000
mean      5312.172769
std       2180.393838
min       1938.312067
25%       3179.374015
50%       5213.952000
75%       7459.699750
max       9180.728000
Name: TotalCharge, dtype: float64
```

[ ]:
```
df.TotalCharge.nlargest(n=20)
```

[ ]:
```
CaseOrder
5713    9180.728
8988    9169.248
5501    9080.912
9404    9077.388
5462    9067.605
7711    9065.054
9136    9028.118
7826    9022.166
9158    9012.388
7295    9004.401
7147    8990.888
9257    8979.087
7215    8975.566
7044    8969.073
```

```
9583    8963.124
6690    8962.874
5649    8957.794
7613    8950.139
5066    8948.540
8916    8943.753
Name: TotalCharge, dtype: float64
```

All of the previous statistics describe the independent variables, which we are testing for a relationship with the days spent hospitalized. It is notable that there are no null values in this column, as there were in D206, so nothing here needs fixed in that regard. No hospitalization of under 1 day is counted, which is a potential limitation in our data to be kept in mind that this reflects only patients who stayed at least 24 hours in the hospital, ignoring patients who were discharged more quickly.

Within this dataset, the mean number of days spent hospitalized is 34, but the standard deviation is very large, at 26 days. The data as a whole ranges from 1 day to 72 days. This puts the minimum at 1.27 standard deviations from the mean, while the maximum is 1.5 standard deviations from the mean. This indicates that the distribution is skewed a bit to one side. Again, a quick check of the largest values for this variable shows that the maximum of ~72 days is not an isolated outlier.

[ ]:
```
df.Initial_days.describe()
```

[ ]:
```
count    10000.000000
mean        34.455299
std         26.309341
min          1.001981
25%          7.896215
50%         35.836244
75%         61.161020
max         71.981490
Name: Initial_days, dtype: float64
```

[ ]:
```
df.Initial_days.nlargest(n=20)
```

[ ]:
```
CaseOrder
7969    71.98149
5327    71.96869
7480    71.96546
6167    71.96415
8067    71.96342
5875    71.96164
5830    71.96134
9160    71.95813
8818    71.95472
7525    71.94732
9075    71.94459
7840    71.92930
```

```
9678    71.92647
9222    71.92413
5163    71.92171
9102    71.90712
9767    71.90694
5375    71.90056
6602    71.89863
7215    71.89805
Name: Initial_days, dtype: float64
```

C3: Preparation of Data

As explained in more detail above in the data prep goals and necessary manipulation section, the data will be cleaned using some of the same code that I used in D206, though some elements are modified. Zip codes are returned back to being strings, rather than integers. Time zones are re-standardized to the 9 US time zones. Categorical columns are cast to categorical, rather than being string objects. Those categorical columns which are boolean in nature are instead remapped to integer (1 = True/Yes, 0 = False/No). The currency columns will be rounded to more accurately reflect the handling of US currency (not to 6 decimals of precision). Finally, the survey response columns will be cast as an ordered categorical datatype, though they will be modified to reverse the scale, such that 1 < 8, rather than the current process of 1 > 8. Finally, the categorical datatypes being used for the multiple regression analysis will be "dummied" using one-hot encoding. For ease of use, a new dataframe will be constructed with all of the necessary columns included for the regression, omitting the ~40 columns which are not needed.

[ ]:
```
# Convert column to string from int, then front-fill string with 0's to reach 5␣
 ↪chars
df['Zip'] = df['Zip'].astype("str").str.zfill(5)
# Convert column to category from string
df["Area"] = df["Area"].astype("category")
# Replace city-specific values with time-zone specific values
df.TimeZone.replace({
    # Puerto Rico does not observe DST, stays on Atlantic Standard Time all␣
 ↪year long
    "America/Puerto_Rico" : "US - Puerto Rico",
    # US - Eastern observes DST
    "America/New_York": "US - Eastern",
    "America/Detroit" : "US - Eastern",
    "America/Indiana/Indianapolis" : "US - Eastern",
    "America/Indiana/Vevay" : "US - Eastern",
    "America/Indiana/Vincennes" : "US - Eastern",
    "America/Kentucky/Louisville" : "US - Eastern",
    "America/Toronto" : "US - Eastern",
    "America/Indiana/Marengo" : "US - Eastern",
    "America/Indiana/Winamac" : "US - Eastern",
    # US - Central observes DST
    "America/Chicago" : "US - Central",
    "America/Menominee" : "US - Central",
```

```
    "America/Indiana/Knox" : "US - Central",
    "America/Indiana/Tell_City" : "US - Central",
    "America/North_Dakota/Beulah" : "US - Central",
    "America/North_Dakota/New_Salem" : "US - Central",
    # US - Mountain observes DST
    "America/Denver" : "US - Mountain",
    "America/Boise" : "US - Mountain",
    # Arizona does not observe DST, stays on Mountain Standard Time all year␣
 ↪long
    "America/Phoenix" : "US - Arizona",
    # US - Pacific observes DST
    "America/Los_Angeles" : "US - Pacific",
    # US - Alaskan observes DST
    "America/Nome" : "US - Alaskan",
    "America/Anchorage" : "US - Alaskan",
    "America/Sitka" : "US - Alaskan",
    "America/Yakutat" : "US - Alaskan",
    # US - Aleutian observes DST
    "America/Adak" : "US - Aleutian",
    # US - Hawaiian does not observe DST, stays on Hawaii Standard Time all year
    "Pacific/Honolulu" : "US - Hawaiian"
    }, inplace=True)
# Convert column to category from string
df["TimeZone"] = df["TimeZone"].astype("category")
# Reformat column representing currency in USD to 3 decimal places from 6
df["Income"] = df["Income"].astype(int)
# Convert column to category from string
df["Marital"] = df["Marital"].astype("category")
# Convert column to category from string
df["Gender"] = df["Gender"].astype("category")
# Recast object > boolean wants to turn everything True, need to map Yes/No to␣
 ↪True/False
bool_mapping = {"Yes" : 1, "No" : 0}
# Convert column to boolean from string
df["ReAdmis"] = df["ReAdmis"].map(bool_mapping)
# Convert column to boolean from string
df["Soft_drink"] = df["Soft_drink"].map(bool_mapping)
# Convert column to category from string
df["Initial_admin"] = df["Initial_admin"].astype("category")
# Convert column to boolean from string
df["HighBlood"] = df["HighBlood"].map(bool_mapping)
# Convert column to boolean from string
df["Stroke"] = df["Stroke"].map(bool_mapping)
# Convert column to category from string
df["Complication_risk"] = df["Complication_risk"].astype("category")
# Convert column to boolean from string
df["Overweight"] = df["Overweight"].map(bool_mapping)
```

```python
# Convert column to boolean from string
df["Arthritis"] = df["Arthritis"].map(bool_mapping)
# Convert column to boolean from string
df["Diabetes"] = df["Diabetes"].map(bool_mapping)
# Convert column to boolean from string
df["Hyperlipidemia"] = df["Hyperlipidemia"].map(bool_mapping)
# Convert column to boolean from string
df["BackPain"] = df["BackPain"].map(bool_mapping)
# Convert column to boolean from string
df["Anxiety"] = df["Anxiety"].map(bool_mapping)
# Convert column to boolean from string
df["Allergic_rhinitis"] = df["Allergic_rhinitis"].map(bool_mapping)
# Convert column to boolean from string
df["Reflux_esophagitis"] = df["Reflux_esophagitis"].map(bool_mapping)
# Convert column to boolean from string
df["Asthma"] = df["Asthma"].map(bool_mapping)
# Convert column to category from string
df["Services"] = df["Services"].astype("category")
# Reformat column representing currency in USD to 3 decimal places from 6
df["TotalCharge"] = df.TotalCharge.round(3)
# Reformat column representing currency in USD to 3 decimal places from 6
df["Additional_charges"] = df.Additional_charges.round(3)
# Establish map for reversing survey questions to reflect a truth where 1 < 8 (currently the reverse)
survey_mapping = {1: 8, 2: 7, 3 : 6, 4: 5, 5: 4, 6: 3, 7 : 2, 8 : 1}
# Establish ordered categorical datatype structure ("1" < "2" < ... < "7" < "8") for survey response columns
survey_scores = CategoricalDtype(categories=["1", "2", "3", "4", "5", "6", "7", "8"], ordered=True)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item1"] = df["Item1"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical will act up without this)
df["Item1"] = df["Item1"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item1"] = df["Item1"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item2"] = df["Item2"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical will act up without this)
df["Item2"] = df["Item2"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item2"] = df["Item2"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item3"] = df["Item3"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical will act up without this)
```

```
df["Item3"] = df["Item3"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item3"] = df["Item3"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item4"] = df["Item4"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical␣
↪will act up without this)
df["Item4"] = df["Item4"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item4"] = df["Item4"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item5"] = df["Item5"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical␣
↪will act up without this)
df["Item5"] = df["Item5"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item5"] = df["Item5"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item6"] = df["Item6"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical␣
↪will act up without this)
df["Item6"] = df["Item6"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item6"] = df["Item6"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item7"] = df["Item7"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical␣
↪will act up without this)
df["Item7"] = df["Item7"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item7"] = df["Item7"].astype(survey_scores)
# Remap column to reflect 1 < 8, rather than 1 > 8
df["Item8"] = df["Item8"].map(survey_mapping)
# Map integers to be strings instead (conversion from int > ordered categorical␣
↪will act up without this)
df["Item8"] = df["Item8"].map(str)
# Reassign datatype from strings to created survey_scores datatype
df["Item8"] = df["Item8"].astype(survey_scores)
# Generate columns of dummy values for dataframe's Gender column
gender_temp_df = pd.get_dummies(data=df["Gender"], drop_first=True)
# Generate columns of dummy values for dataframe's Initial_admin column
initial_admit_temp_df = pd.get_dummies(data=df["Initial_admin"],␣
↪drop_first=True)
# Generate columns of dummy values for dataframe's Complication_risk column
comp_risk_temp_df = pd.get_dummies(data=df["Complication_risk"],␣
↪drop_first=True)
```

```python
# Create new df with only the variables we're interested in
regress_df = df[["Children", "Age", "Income", "VitD_levels", "Doc_visits",␣
↪"Arthritis", "Diabetes", "BackPain", "Initial_days", "TotalCharge"]]
# Generate and apply Pythonic names because the non-Pythonic names annoy me
pythonic_columns = ["num_children", "age", "income", "vit_d_level",␣
↪"dr_visits", "arthritis", "diabetes", "back_pain", "days_hospitalized",␣
↪"avg_daily_charge"]
regress_df.set_axis(pythonic_columns, axis=1, inplace=True)
# Insert the generated dummy variables to new dataframe, placing them in the␣
↪same order as the original dataframe
# Dummies for Complication Risk
regress_df.insert(5, "comp_risk_medium", comp_risk_temp_df.Medium)
regress_df.insert(5, "comp_risk_low", comp_risk_temp_df.Low)
# Dummies for Initial Admit
regress_df.insert(5, "initial_admit_emerg", initial_admit_temp_df["Emergency␣
↪Admission"])
regress_df.insert(5, "initial_admit_observ", initial_admit_temp_df["Observation␣
↪Admission"])
# Dummies for Gender
regress_df.insert(3, "gender_nonbinary", gender_temp_df.Male)
regress_df.insert(3, "gender_male", gender_temp_df.Male)
# Check resulting dataframe
regress_df
```

[ ]:

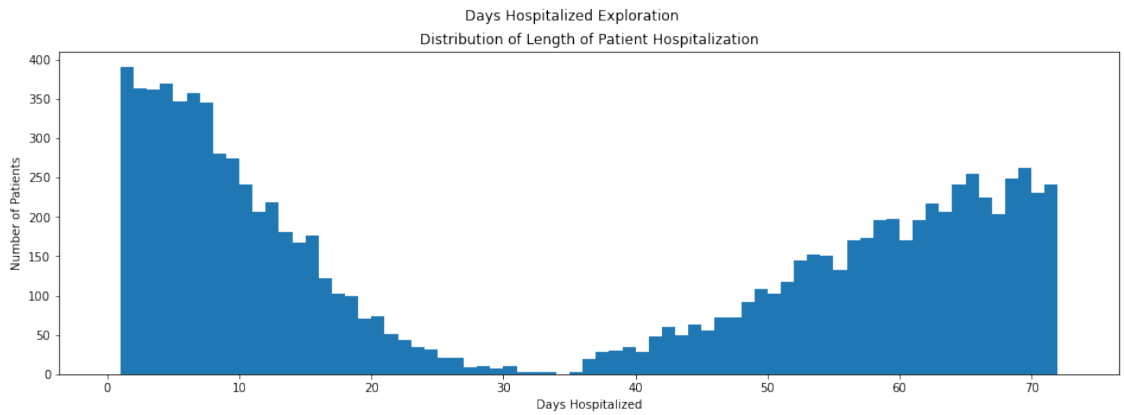
```
           num_children  age  income  gender_male  gender_nonbinary  \
CaseOrder
1                     1   53   86575            1                 1
2                     3   51   46805            0                 0
3                     3   53   14370            0                 0
4                     0   78   39741            1                 1
5                     1   22    1209            0                 0
…                   …   …     …            …                 …
9996                  2   25   45967            1                 1
9997                  4   87   14983            1                 1
9998                  3   45   65917            0                 0
9999                  3   43   29702            1                 1
10000                 8   70   62682            0                 0

           vit_d_level  dr_visits  initial_admit_observ  initial_admit_emerg  \
CaseOrder
1            19.141466          6                     0                    1
2            18.940352          4                     0                    1
3            18.057507          4                     0                    0
4            16.576858          4                     0                    0
5            17.439069          5                     0                    0
…                  …        …                     …                    …
9996         16.980860          4                     0                    1
```

```
9997         18.177020         5                    0                    0
9998         17.129070         4                    0                    0
9999         19.910430         5                    0                    1
10000        18.388620         5                    1                    0

           comp_risk_low  comp_risk_medium  arthritis  diabetes  back_pain  \
CaseOrder
1                      0                 1          1         1          1
2                      0                 0          0         0          0
3                      0                 1          0         1          0
4                      0                 1          1         0          0
5                      1                 0          0         0          0
…                    …               …       …        …       …
9996                   0                 1          0         0          0
9997                   0                 1          1         1          0
9998                   0                 0          0         0          0
9999                   0                 1          0         0          1
10000                  1                 0          1         0          0

           days_hospitalized  avg_daily_charge
CaseOrder
1                  10.585770          3726.703
2                  15.129562          4193.190
3                   4.772177          2434.234
4                   1.714879          2127.830
5                   1.254807          2113.073
…                        …                 …
9996               51.561220          6850.942
9997               68.668240          7741.690
9998               70.154180          8276.481
9999               63.356900          7644.483
10000              70.850590          7887.553

[10000 rows x 16 columns]
```

C4: Univariate and Bivariate Distributions

Before examining the explanatory variables, it seems prudent to examine the dependent variable first.

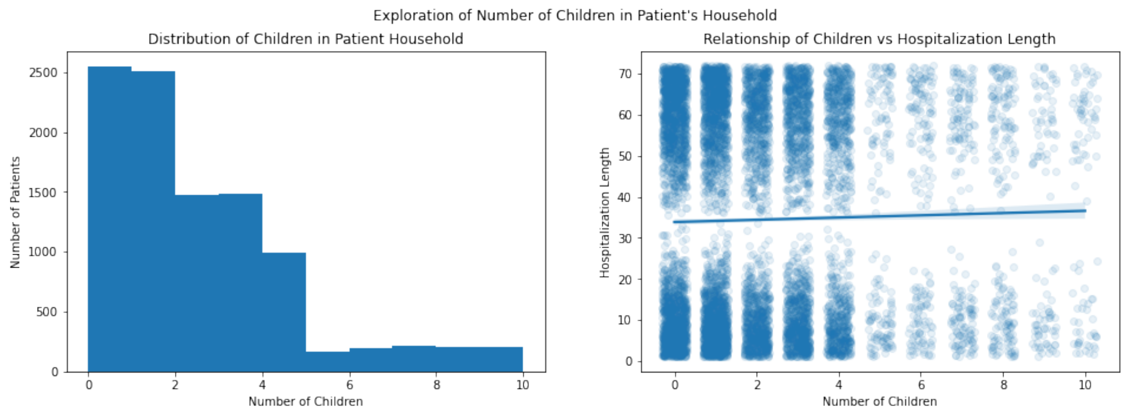
[ ]:
```python
plt.figure(figsize = [16,5])
plt.suptitle("Days Hospitalized Exploration")
plt.title("Distribution of Length of Patient Hospitalization")
bins = np.arange(0, 74, 1)
plt.hist(data=regress_df, x="days_hospitalized", bins=bins)
plt.xlabel("Days Hospitalized")
plt.ylabel("Number of Patients");
```



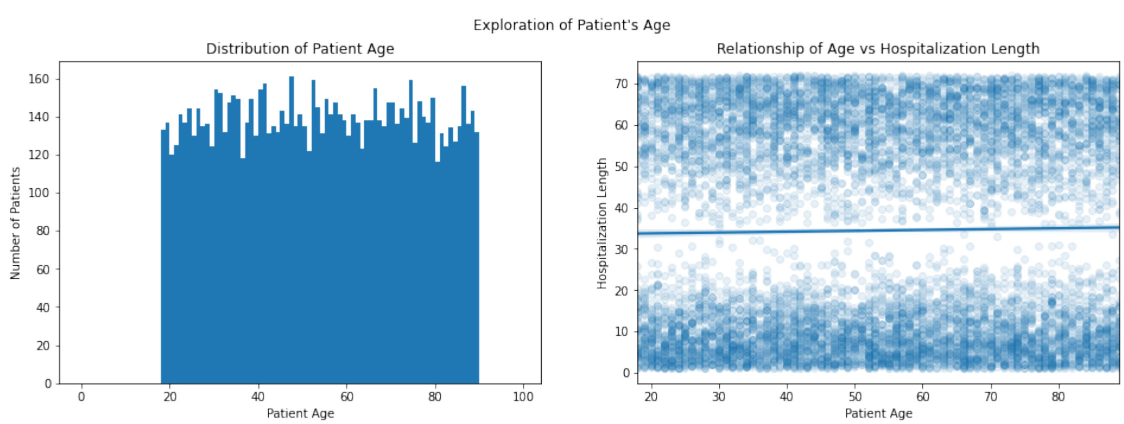
This is a surprising outcome - the summary statistics led me to assume a somewhat normal distribution would be seen here, with a right skew. I was not expecting to find this distribution with two peaks and the mean actually existing at/near a trough.

The remaining variables will be examined both in their distribution alone and in their relationship to the days hospitalized variable.

[ ]:
```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Number of Children in Patient's Household")

# LEFT plot: Univariate exploration of num_children
plt.subplot(1, 2, 1)
plt.title('Distribution of Children in Patient Household')
bins = np.arange(0, regress_df.num_children.max() + 1, 1)
plt.hist(data=regress_df, x="num_children", bins=bins)
plt.xlabel('Number of Children')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of num_children vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Children vs Hospitalization Length")
sns.regplot(data=regress_df, x="num_children", y="days_hospitalized",␣
 ↪x_jitter=0.3, scatter_kws={'alpha' : 1/10})
plt.xlabel("Number of Children")
plt.ylabel("Hospitalization Length");
```

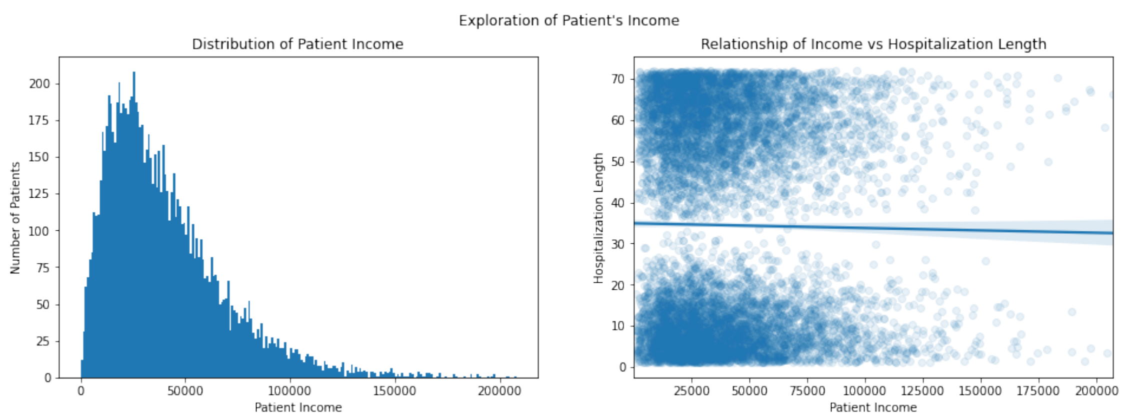


[ ]:
```python
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient's Age")

# LEFT plot: Univariate exploration of age
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Age')
bins = np.arange(0, 100, 1)
plt.hist(data=regress_df, x="age", bins=bins)
plt.xlabel('Patient Age')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of age vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Age vs Hospitalization Length")
sns.regplot(data=regress_df, x="age", y="days_hospitalized",␣
 ↪scatter_kws={'alpha' : 1/10})
plt.xlabel("Patient Age")
plt.ylabel("Hospitalization Length");
```



[ ]:

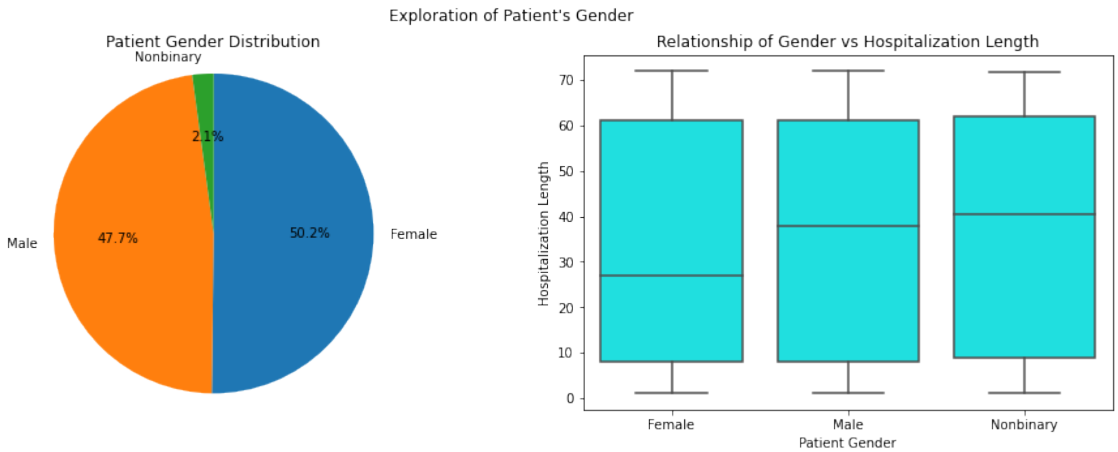
```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient's Income")

# LEFT plot: Univariate exploration of income
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Income')
bins = np.arange(0, regress_df.income.max() + 1000, 1000)
plt.hist(data=regress_df, x="income", bins=bins)
plt.xlabel('Patient Income')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of income vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Income vs Hospitalization Length")
sns.regplot(data=regress_df, x="income", y="days_hospitalized",␣
↪scatter_kws={'alpha' : 1/10})
plt.xlabel("Patient Income")
plt.ylabel("Hospitalization Length");
```



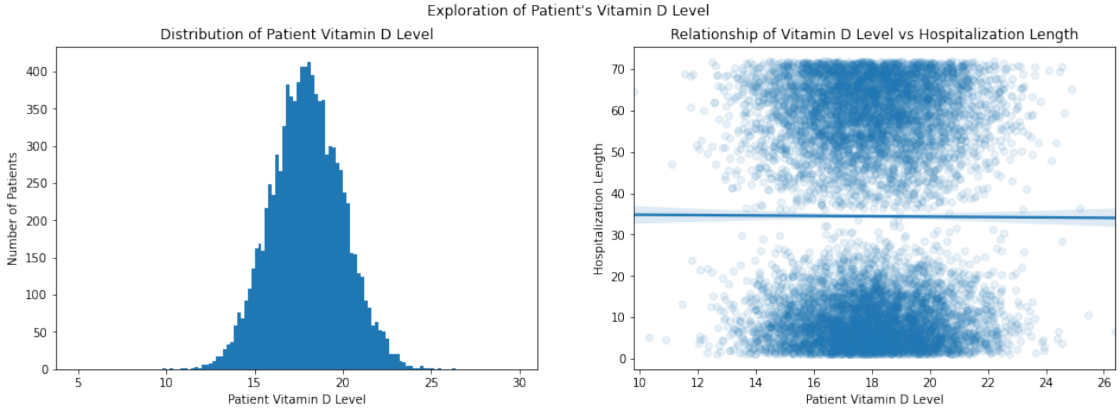
[ ]:

```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient's Gender")

# LEFT plot: Univariate exploration of gender
plt.subplot(1, 2, 1)
plt.title("Patient Gender Distribution")
gender_counts = df["Gender"].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',␣
↪startangle=90, counterclock = False)
plt.axis('square');
```

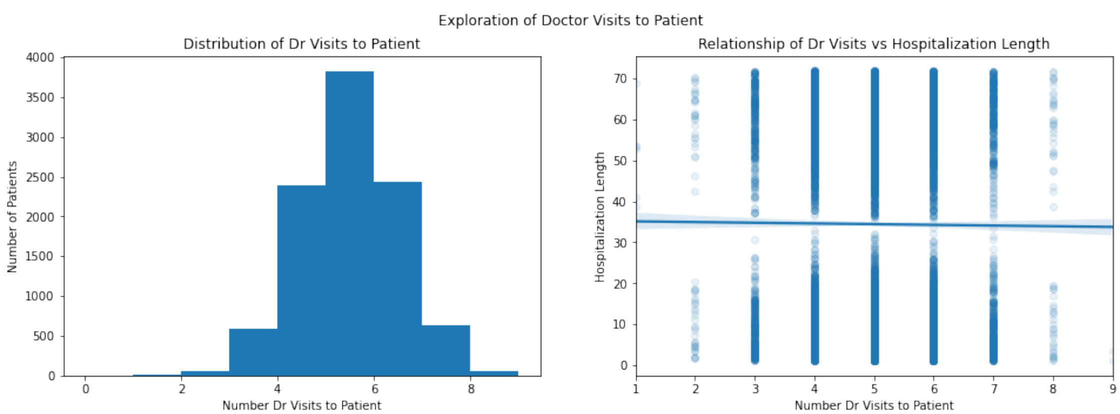
```python
# RIGHT plot: Bivariate exploration of gender vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Gender vs Hospitalization Length")
sns.boxplot(data=df, x="Gender", y='Initial_days', color="cyan")
plt.xlabel("Patient Gender")
plt.ylabel("Hospitalization Length");
```



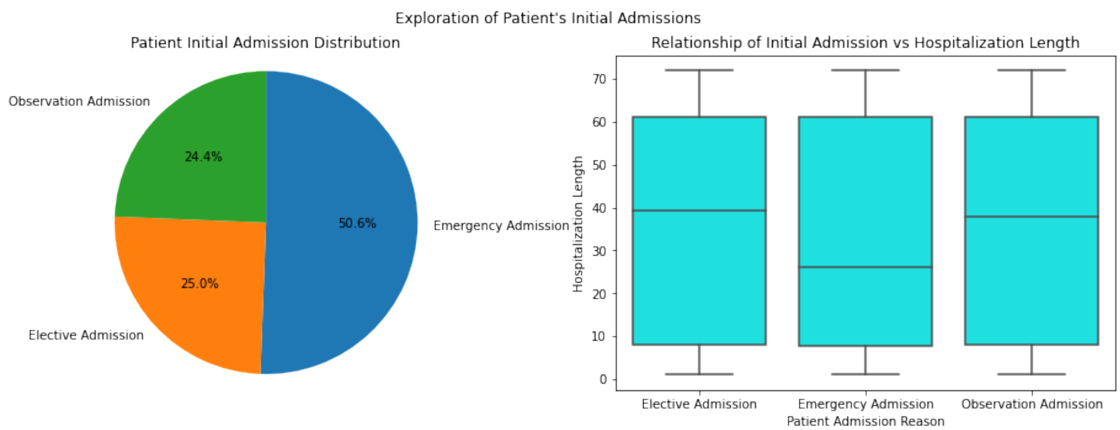
[ ]:
```python
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient's Vitamin D Level")

# LEFT plot: Univariate exploration of vit_d_level
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Vitamin D Level')
bins = np.arange(5, 30, 0.20)
plt.hist(data=regress_df, x="vit_d_level", bins=bins)
plt.xlabel('Patient Vitamin D Level')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of vit_d_level vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Vitamin D Level vs Hospitalization Length")
sns.regplot(data=regress_df, x="vit_d_level", y="days_hospitalized",
 ↪scatter_kws={'alpha' : 1/10})
plt.xlabel("Patient Vitamin D Level")
plt.ylabel("Hospitalization Length");
```




```
[ ]: plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Doctor Visits to Patient")

# LEFT plot: Univariate exploration of dr_visits
plt.subplot(1, 2, 1)
plt.title('Distribution of Dr Visits to Patient')
bins = np.arange(0, 10, 1)
plt.hist(data=regress_df, x="dr_visits", bins=bins)
plt.xlabel('Number Dr Visits to Patient')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of dr_visits vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Dr Visits vs Hospitalization Length")
sns.regplot(data=regress_df, x="dr_visits", y="days_hospitalized",␣
↪scatter_kws={'alpha' : 1/10})
plt.xlabel("Number Dr Visits to Patient")
plt.ylabel("Hospitalization Length");
```

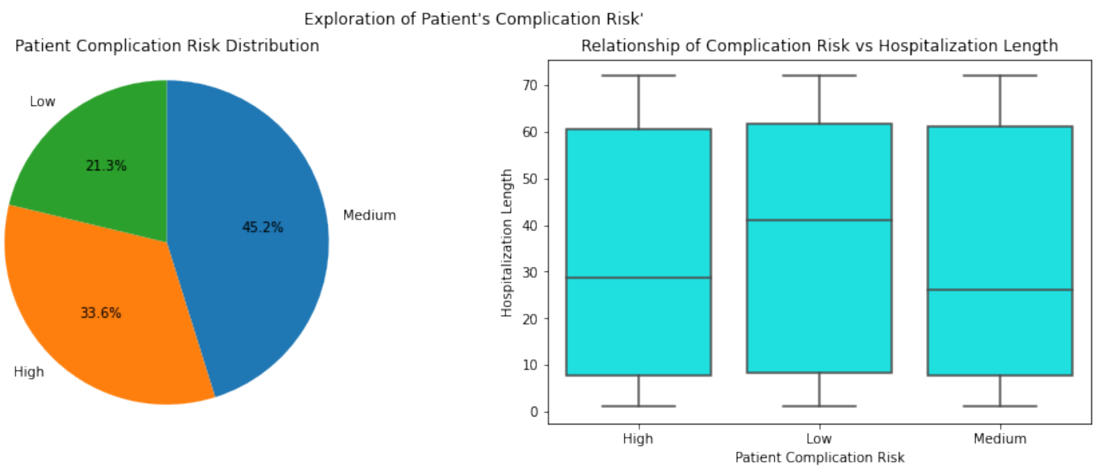


[ ]:
```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient's Initial Admissions")

# LEFT plot: Univariate exploration of initial_admin
plt.subplot(1, 2, 1)
plt.title("Patient Initial Admission Distribution")
init_admit_counts = df["Initial_admin"].value_counts()
plt.pie(init_admit_counts, labels=init_admit_counts.index, autopct='%1.1f%%',␣
↪startangle=90, counterclock = False)
plt.axis('square');

# RIGHT plot: Bivariate exploration of Initial_admin vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Initial Admission vs Hospitalization Length")
sns.boxplot(data=df, x="Initial_admin", y='Initial_days', color="cyan")
plt.xlabel("Patient Admission Reason")
plt.ylabel("Hospitalization Length");
```

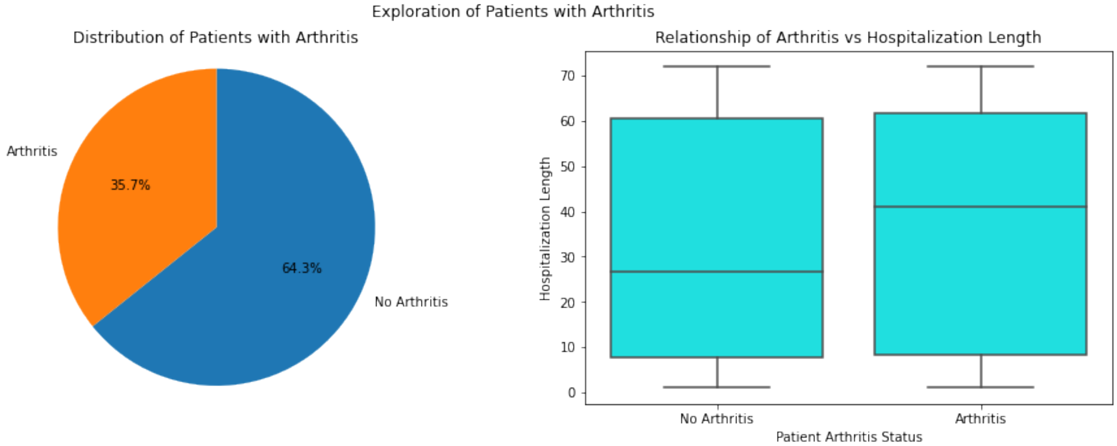


[ ]:
```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient's Complication Risk'")

# LEFT plot: Univariate exploration of complication_risk
plt.subplot(1, 2, 1)
plt.title("Patient Complication Risk Distribution")
comp_risk_counts = df["Complication_risk"].value_counts()
comp_risk_labels = ["Medium", "High", "Low"]
plt.pie(comp_risk_counts, labels=comp_risk_counts.index, autopct='%1.1f%%',␣
↪startangle=90, counterclock = False)
plt.axis('square');
```

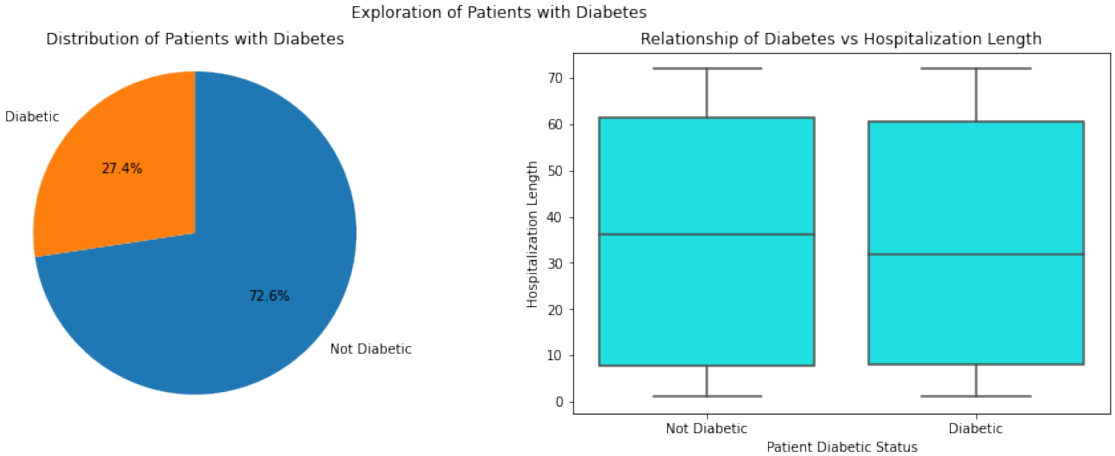
```
# RIGHT plot: Bivariate exploration of complication_risk vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Complication Risk vs Hospitalization Length")
sns.boxplot(data=df, x="Complication_risk", y='Initial_days', color="cyan")
plt.xlabel("Patient Complication Risk")
plt.ylabel("Hospitalization Length");
```



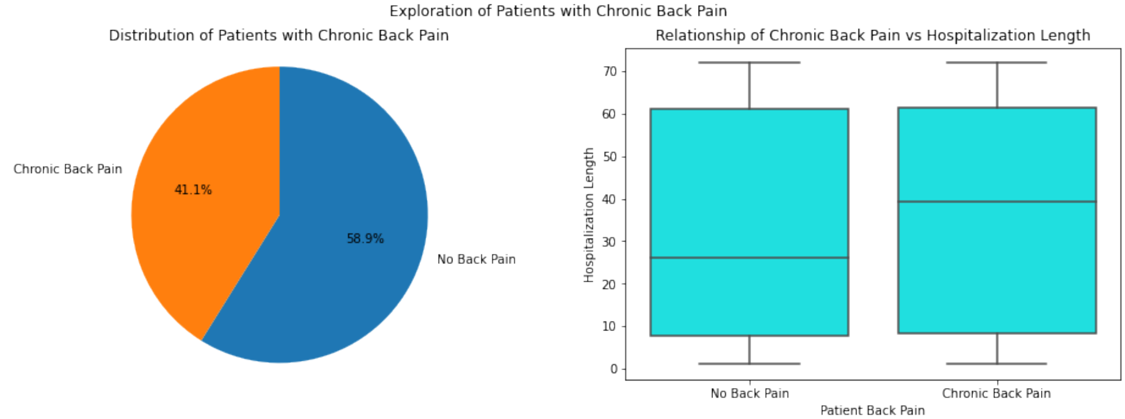
[ ]:
```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patients with Arthritis")

# LEFT plot: Univariate exploration of arthritis
plt.subplot(1, 2, 1)
plt.title('Distribution of Patients with Arthritis')
arthritis_counts = regress_df.arthritis.value_counts()
arthritis_labels = ["No Arthritis", "Arthritis"]
plt.pie(arthritis_counts, labels=arthritis_labels, autopct='%1.1f%%',
    startangle=90, counterclock=False)
plt.axis('square');

# RIGHT plot: Bivariate exploration of arthritis vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Arthritis vs Hospitalization Length")
sns.boxplot(data=regress_df, x="arthritis", y='days_hospitalized', color="cyan")
plt.xticks(ticks=[0,1], labels = arthritis_labels)
plt.xlabel("Patient Arthritis Status")
plt.ylabel("Hospitalization Length");
```



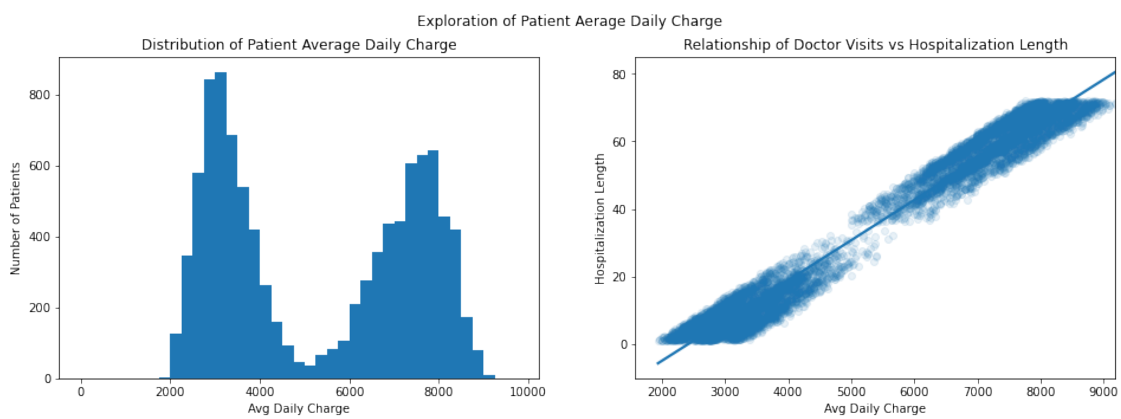
[ ]:
```python
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patients with Diabetes")

# LEFT plot: Univariate exploration of diabetes
plt.subplot(1, 2, 1)
plt.title('Distribution of Patients with Diabetes')
diabetes_counts = regress_df.diabetes.value_counts()
diabetes_labels = ["Not Diabetic", "Diabetic"]
plt.pie(diabetes_counts, labels=diabetes_labels, autopct='%1.1f%%',␣
↪startangle=90, counterclock=False)
plt.axis('square');

# RIGHT plot: Bivariate exploration of diabetes vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Diabetes vs Hospitalization Length")
sns.boxplot(data=regress_df, x="diabetes", y='days_hospitalized', color="cyan")
plt.xticks(ticks=[0,1], labels = diabetes_labels)
plt.xlabel("Patient Diabetic Status")
plt.ylabel("Hospitalization Length");
```



[ ]:
```
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patients with Chronic Back Pain")

# LEFT plot: Univariate exploration of back_pain
plt.subplot(1, 2, 1)
plt.title('Distribution of Patients with Chronic Back Pain')
back_pain_counts = regress_df.back_pain.value_counts()
back_pain_labels = ["No Back Pain", "Chronic Back Pain"]
plt.pie(back_pain_counts, labels=back_pain_labels, autopct='%1.1f%%',
 ↪startangle=90, counterclock=False)
plt.axis('square');

# RIGHT plot: Bivariate exploration of back_pain vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Chronic Back Pain vs Hospitalization Length")
sns.boxplot(data=regress_df, x="back_pain", y='days_hospitalized', color="cyan")
plt.xticks(ticks=[0,1], labels = back_pain_labels)
plt.xlabel("Patient Back Pain")
plt.ylabel("Hospitalization Length");
```




```
[ ]: plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Patient Aerage Daily Charge")

# LEFT plot: Univariate exploration of avg_daily_charge
plt.subplot(1, 2, 1)
plt.title('Distribution of Patient Average Daily Charge')
bins = np.arange(0, 10000, 250)
plt.hist(data=regress_df, x="avg_daily_charge", bins=bins)
plt.xlabel('Avg Daily Charge')
plt.ylabel("Number of Patients");

# RIGHT plot: Bivariate exploration of avg_daily_charge vs days_hospitalized
plt.subplot(1, 2, 2)
plt.title("Relationship of Doctor Visits vs Hospitalization Length")
sns.regplot(data=regress_df, x="avg_daily_charge", y="days_hospitalized",␣
↪x_jitter=0.3, scatter_kws={'alpha' : 1/10})
plt.xlabel("Avg Daily Charge")
plt.ylabel("Hospitalization Length");
```



C5: Copy of Prepared Data Set

A copy of the prepared dataset is submitted alongside this analysis. The full cleaned dataframe can be found in full_clean.csv, while the reduced dataframe, containing only the variables I want to analyze (including dummies for the categorical columns) is in red_clean.csv. I've submitted both, as the rubric wasn't really clear on which it would be interested in.

[ ]:
```
# Save dataframe to CSV, ignore index (if included, this will create an␣
↪additional unnecessary column)
df.to_csv('task1_full_clean.csv', index=False)

# Save dataframe to CSV, ignore index (if included, this will create an␣
↪additional unnecessary column)
regress_df.to_csv('task1_red_clean.csv', index=False)
```

D1: Initial Multiple Regression Model

I constructed an initial multiple regression model below, including all of the predictor variables that I identified in section C2. This model was generated with a y-intercept, by using the `.assign(const=1)`function while generating my X variables. This code was generated with assistance from Mark Keith's Machine Learning in Python course materials on YouTube. This multiple regression model will be reduced to remove any issues and focus only on variables which contribute meaningfully to the explanatory variable.

[ ]:
```
# Initial model for Multiple Linear Regression

# Set dependent variable
y = regress_df.days_hospitalized
# Set multiple independent variables
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "vit_d_level", "dr_visits", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 3.986e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:03   Log-Likelihood:                -15249.
No. Observations:               10000   AIC:                         3.053e+04
Df Residuals:                    9985   BIC:                         3.064e+04
Df Model:                          14
```

```
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children             0.0012      0.005      0.229      0.819      -0.009
0.011
age                     -0.0003      0.001     -0.561      0.575      -0.001
0.001
income               -3.601e-07    3.9e-07     -0.922      0.356   -1.13e-06
4.05e-07
gender_male             -0.0006      0.011     -0.057      0.955      -0.022
0.021
gender_nonbinary        -0.0006      0.011     -0.057      0.955      -0.022
0.021
vit_d_level              0.0029      0.006      0.525      0.600      -0.008
0.014
dr_visits                0.0096      0.011      0.899      0.369      -0.011
0.030
initial_admit_observ     0.0202      0.032      0.637      0.524      -0.042
0.082
initial_admit_emerg     -6.2641      0.027   -229.353      0.000      -6.318
-6.211
comp_risk_low            5.0828      0.031    164.481      0.000       5.022
5.143
comp_risk_medium         5.0335      0.025    197.661      0.000       4.984
5.083
arthritis               -0.9072      0.023    -39.028      0.000      -0.953
-0.862
diabetes                -0.9179      0.025    -36.750      0.000      -0.967
-0.869
back_pain               -1.0633      0.023    -46.947      0.000      -1.108
-1.019
avg_daily_charge         0.0122   5.16e-06   2359.774      0.000       0.012
0.012
const                  -29.4951      0.124   -237.141      0.000     -29.739
-29.251
==============================================================================
Omnibus:                      195.513   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              165.223
Skew:                          -0.250   Prob(JB):                     1.33e-36
Kurtosis:                       2.618   Cond. No.                     3.47e+20
==============================================================================

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

```
specified.
[2] The smallest eigenvalue is 2.06e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[ ]:
```
results.resid.std(ddof=X.shape[1])
```

[ ]:
```
1.1126862546385587
```

D2: Reduction Justification

We can see in the above multiple regression model that we receive a warning that there are possibly strong multicolinearity problems with our model. One of the key assumptions of multiple regression is that our independent variables do not have a high multicolinearity, so we should check for this using the Variance Inflation Factor (VIF). Any factors with a VIF larger than 10 should be removed due to high multicolinearity, but this analysis has to be repeated after each factor is removed. Thus, I will run this analysis with the following code (WGU Courseware Resources, remove the factor with the largest VIF, and then rerun the analysis, repeating until all VIF scores are below 10.

[ ]:
```
# Warning exists for multicolinearity - check for VIF to see if variables␣
↪should be eliminated due to high multicolinearity
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "vit_d_level", "dr_visits", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]]

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                 feature        VIF
0           num_children   1.932741
1                    age   7.361227
2                 income   2.963630
3            gender_male        inf
4       gender_nonbinary        inf
5            vit_d_level  29.950450
6              dr_visits  19.835725
7   initial_admit_observ   1.950991
8    initial_admit_emerg   3.014566
9          comp_risk_low   1.618548
10      comp_risk_medium   2.324442
11             arthritis   1.555091
12              diabetes   1.373587
13             back_pain   1.697394
14      avg_daily_charge   6.760867
```

```
C:\Users\hasek\anaconda3\lib\site-
packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by
zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

[ ]:
```python
# Eliminated vit_d_level (VIF = 29.95), rerunning analysis to see if any VIF␣
↪still above 10
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "dr_visits", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]]

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                 feature        VIF
0           num_children   1.909810
1                    age   6.650950
2                 income   2.888846
3            gender_male        inf
4       gender_nonbinary        inf
5              dr_visits  12.338329
6   initial_admit_observ   1.904371
7    initial_admit_emerg   2.919809
8          comp_risk_low   1.587692
9       comp_risk_medium   2.244750
10             arthritis   1.547064
11              diabetes   1.370846
12             back_pain   1.688951
13      avg_daily_charge   6.218787
```

```
C:\Users\hasek\anaconda3\lib\site-
packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by
zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

[ ]:
```python
# Eliminated dr_visits (VIF = 12.34), rerunning analysis to see if any VIF␣
↪still above 10
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "initial_admit_observ", "initial_admit_emerg",␣
↪"comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
↪"avg_daily_charge"]]
```

```
vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                 feature       VIF
0            num_children  1.880773
1                     age  5.568718
2                  income  2.740994
3             gender_male       inf
4        gender_nonbinary       inf
5    initial_admit_observ  1.810973
6     initial_admit_emerg  2.769873
7           comp_risk_low  1.548161
8        comp_risk_medium  2.148653
9               arthritis  1.535005
10               diabetes  1.358931
11              back_pain  1.670107
12       avg_daily_charge  5.435098
```

```
C:\Users\hasek\anaconda3\lib\site-
packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by
zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

Another useful thing to do with this data is to transform it. In the initial model, it is difficult to meaningfully compare x variables with each other because they are all on different scales - age runs from 18 - 89, income runs from \$150 to over \$200,000, etc. One way to address this to more easily allow for comparison of the explanatory variables is to normalize everything based upon the min and max, where a column is recast such that the minimum value = 0 and the maximum value = 1. The code for this process came from another of Mark Keith's series of videos for Machine Learning in Python on YouTube.

[ ]:
```
reg_df_minmax = pd.DataFrame(preprocessing.MinMaxScaler().
    ↪fit_transform(regress_df), columns=regress_df.columns)
reg_df_minmax
```

[ ]:
```
      num_children       age    income  gender_male  gender_nonbinary  \
0              0.1  0.492958  0.417301          1.0               1.0
1              0.3  0.464789  0.225264          0.0               0.0
2              0.3  0.492958  0.068645          0.0               0.0
3              0.0  0.845070  0.191154          1.0               1.0
4              0.1  0.056338  0.005094          0.0               0.0
…              …         …         …            …                 …
9995           0.2  0.098592  0.221217          1.0               1.0
```

```
9996          0.4  0.971831  0.071605         1.0               1.0
9997          0.3  0.380282  0.317550         0.0               0.0
9998          0.3  0.352113  0.142678         1.0               1.0
9999          0.8  0.732394  0.301929         0.0               0.0

      vit_d_level  dr_visits  initial_admit_observ  initial_admit_emerg  \
0        0.562756      0.625                   0.0                  1.0
1        0.550632      0.375                   0.0                  1.0
2        0.497410      0.375                   0.0                  0.0
3        0.408150      0.375                   0.0                  0.0
4        0.460128      0.500                   0.0                  0.0
…             …         …                     …                    …
9995     0.432505      0.375                   0.0                  1.0
9996     0.504615      0.500                   0.0                  0.0
9997     0.441440      0.375                   0.0                  0.0
9998     0.609113      0.500                   0.0                  1.0
9999     0.517371      0.500                   1.0                  0.0

      comp_risk_low  comp_risk_medium  arthritis  diabetes  back_pain  \
0               0.0               1.0        1.0       1.0        1.0
1               0.0               0.0        0.0       0.0        0.0
2               0.0               1.0        0.0       1.0        0.0
3               0.0               1.0        1.0       0.0        0.0
4               1.0               0.0        0.0       0.0        0.0
…             …                 …          …         …         …
9995            0.0               1.0        0.0       0.0        0.0
9996            0.0               1.0        1.0       1.0        0.0
9997            0.0               0.0        0.0       0.0        0.0
9998            0.0               1.0        0.0       0.0        1.0
9999            1.0               0.0        1.0       0.0        0.0

      days_hospitalized  avg_daily_charge
0              0.135022          0.246933
1              0.199037          0.311343
2              0.053117          0.068475
3              0.010044          0.026168
4              0.003562          0.024130
…                  …               …
9995           0.712308          0.678314
9996           0.953321          0.801304
9997           0.974256          0.875146
9998           0.878492          0.787882
9999           0.984067          0.821444

[10000 rows x 16 columns]
```

The next process will be reducing the variables which are not statistically significant to the model. Vitamin D levels and number of doctor visits can already be removed due to the multicolinearity issues identified above. This will be done through a process called Backwards Stepwise Elimination, as detailed by Ashutosh Tripathi (Towards Data Science, 2019).

This will be done by generating the multiple regression model and checking the p-values for each variable. I am interested in statistically significant variables, so I will use a threshold (alpha) of 0.05. Any p-value below this is considered statistically significant, while p-values above are not. The model will be generated, and the independent variable with the highest p-value will be eliminated. The model will then be generated again, and the process repeated until all p-values for included variables are below 0.05.

```
[ ]: # Reduce model by eliminating dr_visits and vit_d_levels due to␣
      ↪multicolinearity issues
     # Continue reducing model by backward elimination, removing the highest p-value␣
      ↪(alpha = 0.05) and re-modelling
     # will eventually be left with only the most important variables

     # BACKWARD ELIMINATION # 1: Seek highest p-value above 0.05 (eliminated␣
      ↪dr_visits, vit_d_levels due to multicolinearity)
     y = reg_df_minmax.days_hospitalized
     X = reg_df_minmax[["num_children", "age", "income", "gender_male",␣
      ↪"gender_nonbinary", "initial_admit_observ", "initial_admit_emerg",␣
      ↪"comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
      ↪"avg_daily_charge"]].assign(const=1)

     model = sm.OLS(y, X)
     results = model.fit()
     print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 4.651e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:04   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.472e+04
Df Residuals:                    9987   BIC:                        -5.463e+04
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children             0.0002      0.001      0.232      0.817      -0.001
0.002
age                     -0.0003      0.001     -0.549      0.583      -0.001
```

```
0.001
income                  -0.0010      0.001     -0.917      0.359      -0.003
0.001
gender_male          -1.072e-05      0.000     -0.068      0.946      -0.000
0.000
gender_nonbinary     -1.072e-05      0.000     -0.068      0.946      -0.000
0.000
initial_admit_observ     0.0003      0.000      0.658      0.510      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.429      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.489      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.684      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.030      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.770      0.000      -0.014
-0.012
back_pain               -0.0150      0.000    -46.950      0.000      -0.016
-0.014
avg_daily_charge         1.2428      0.001   2359.944      0.000       1.242
1.244
const                   -0.0957      0.001   -149.616      0.000      -0.097
-0.094
==============================================================================
Omnibus:                      195.218   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.570
Skew:                          -0.249   Prob(JB):                     1.84e-36
Kurtosis:                       2.617   Cond. No.                     1.75e+17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 1.02e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[ ]:
```
# BACKWARD ELIMINATION # 2: Seek highest p-value above 0.05 (eliminated
 ↪gender_male, p-value of 0.946)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["num_children", "age", "income", "gender_nonbinary",
 ↪"initial_admit_observ", "initial_admit_emerg", "comp_risk_low",
 ↪"comp_risk_medium", "arthritis", "diabetes", "back_pain",
 ↪"avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
```

```python
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 4.651e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:04   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.472e+04
Df Residuals:                    9987   BIC:                        -5.463e+04
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children             0.0002      0.001      0.232      0.817      -0.001
0.002
age                     -0.0003      0.001     -0.549      0.583      -0.001
0.001
income                  -0.0010      0.001     -0.917      0.359      -0.003
0.001
gender_nonbinary     -2.143e-05      0.000     -0.068      0.946      -0.001
0.001
initial_admit_observ     0.0003      0.000      0.658      0.510      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.429      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.489      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.684      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.030      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.770      0.000      -0.014
-0.012
back_pain               -0.0150      0.000    -46.950      0.000      -0.016
-0.014
avg_daily_charge         1.2428      0.001   2359.944      0.000       1.242
1.244
const                   -0.0957      0.001   -149.616      0.000      -0.097
-0.094
==============================================================================
Omnibus:                      195.218   Durbin-Watson:                   1.975
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.570
Skew:                          -0.249   Prob(JB):                     1.84e-36
Kurtosis:                       2.617   Cond. No.                         12.6
==============================================================================

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

[ ]:
```
# BACKWARD ELIMINATION # 3: Seek highest p-value above 0.05 (eliminated␣
 ↪gender_nonbinary, p-value of 0.946)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["num_children", "age", "income", "initial_admit_observ",␣
 ↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
 ↪"diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 5.074e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:04   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.472e+04
Df Residuals:                    9988   BIC:                        -5.464e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children             0.0002      0.001      0.232      0.816      -0.001
0.002
age                     -0.0003      0.001     -0.548      0.584      -0.001
0.001
income                  -0.0010      0.001     -0.917      0.359      -0.003
0.001
initial_admit_observ     0.0003      0.000      0.660      0.509      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.504      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.497      0.000       0.071
```

```
0.072
comp_risk_medium          0.0709      0.000     197.699      0.000       0.070
0.072
arthritis                -0.0128      0.000     -39.033      0.000      -0.013
-0.012
diabetes                 -0.0129      0.000     -36.771      0.000      -0.014
-0.012
back_pain                -0.0150      0.000     -46.957      0.000      -0.016
-0.014
avg_daily_charge          1.2428      0.001    2360.121      0.000       1.242
1.244
const                    -0.0957      0.001    -154.535      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.200   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.541
Skew:                          -0.249   Prob(JB):                     1.86e-36
Kurtosis:                       2.617   Cond. No.                         12.0
==============================================================================

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

[ ]:

```
# BACKWARD ELIMINATION # 4: Seek highest p-value above 0.05 (eliminated␣
↪num_children, p-value of 0.816)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["age", "income", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 5.582e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:04   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9989   BIC:                        -5.465e+04
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
========
```

```
                            coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
age                      -0.0003      0.001     -0.546      0.585      -0.001
0.001
income                   -0.0010      0.001     -0.916      0.360      -0.003
0.001
initial_admit_observ      0.0003      0.000      0.660      0.509      -0.001
0.001
initial_admit_emerg      -0.0882      0.000   -229.516      0.000      -0.089
-0.087
comp_risk_low             0.0716      0.000    164.505      0.000       0.071
0.072
comp_risk_medium          0.0709      0.000    197.710      0.000       0.070
0.072
arthritis                -0.0128      0.000    -39.034      0.000      -0.013
-0.012
diabetes                 -0.0129      0.000    -36.777      0.000      -0.014
-0.012
back_pain                -0.0150      0.000    -46.963      0.000      -0.016
-0.014
avg_daily_charge          1.2428      0.001   2360.835      0.000       1.242
1.244
const                    -0.0957      0.001   -158.610      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.107   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.502
Skew:                          -0.249   Prob(JB):                     1.90e-36
Kurtosis:                       2.617   Cond. No.                         11.9
==============================================================================

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

[ ]:
```
# BACKWARD ELIMINATION # 5: Seek highest p-value above 0.05 (eliminated age,
↪p-value of 0.585)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["income", "initial_admit_observ", "initial_admit_emerg",
↪"comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain",
↪"avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 6.203e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:05   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9990   BIC:                        -5.466e+04
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
income                  -0.0010      0.001     -0.909      0.363      -0.003
0.001
initial_admit_observ     0.0003      0.000      0.667      0.505      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.536      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.512      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.722      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.040      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.780      0.000      -0.014
-0.012
back_pain               -0.0150      0.000    -46.986      0.000      -0.016
-0.014
avg_daily_charge         1.2428      0.001   2361.202      0.000       1.242
1.244
const                   -0.0958      0.001   -177.580      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.297   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.661
Skew:                          -0.249   Prob(JB):                     1.75e-36
Kurtosis:                       2.617   Cond. No.                         11.3
==============================================================================

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

[ ]:
```
# BACKWARD ELIMINATION # 6: Seek highest p-value above 0.05 (eliminated␣
↪initial_admit_observ, p-value of 0.505)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["income", "initial_admit_emerg", "comp_risk_low",␣
↪"comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
↪"avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 6.979e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:05   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9991   BIC:                        -5.466e+04
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
income                 -0.0010      0.001     -0.902      0.367      -0.003
0.001
initial_admit_emerg    -0.0884      0.000   -280.249      0.000      -0.089
-0.088
comp_risk_low           0.0716      0.000    164.548      0.000       0.071
0.072
comp_risk_medium        0.0709      0.000    197.812      0.000       0.070
0.072
arthritis              -0.0128      0.000    -39.042      0.000      -0.013
-0.012
diabetes               -0.0129      0.000    -36.785      0.000      -0.014
-0.012
back_pain              -0.0150      0.000    -46.983      0.000      -0.016
-0.014
avg_daily_charge        1.2428      0.001   2361.289      0.000       1.242
1.244
const                  -0.0957      0.000   -192.771      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.261   Durbin-Watson:                   1.975
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.491
Skew:                          -0.249   Prob(JB):                     1.91e-36
Kurtosis:                       2.616   Cond. No.                         11.1
==============================================================================

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

[ ]:
```
# BACKWARD ELIMINATION # 7: Seek highest p-value above 0.05 (eliminated income,␣
 ↪p-value of 0.367)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["initial_admit_emerg", "comp_risk_low", "comp_risk_medium",␣
 ↪"arthritis", "diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 7.976e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:05   Log-Likelihood:                 27373.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9992   BIC:                        -5.467e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
initial_admit_emerg    -0.0884      0.000   -280.292      0.000      -0.089
-0.088
comp_risk_low           0.0716      0.000    164.548      0.000       0.071
0.072
comp_risk_medium        0.0709      0.000    197.823      0.000       0.070
0.072
arthritis              -0.0128      0.000    -39.038      0.000      -0.013
-0.012
diabetes               -0.0129      0.000    -36.778      0.000      -0.014
-0.012
back_pain              -0.0150      0.000    -46.995      0.000      -0.016
-0.014
```

```
avg_daily_charge        1.2428      0.001    2361.509      0.000       1.242
1.244
const                  -0.0959      0.000    -217.782      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      194.884   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.109
Skew:                          -0.248   Prob(JB):                     2.31e-36
Kurtosis:                       2.616   Cond. No.                         6.09
==============================================================================

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

[ ]:
```python
results.resid.std(ddof=X.shape[1])
```

[ ]:
```
0.01567200935795125
```

The last model ends up being the final model, because every variable has a p-value below 0.05, indicating that it is statistically significant. This reduced model can be compared to our initial model, to verify that this indeed a "better" model. Both models have very high r-squared values and their p-values are identical, so we'll have to use another method to compare the two.

One way to do this is to use the Residual Standard Error, to check the standard error of the resulting residuals under that model. The code for doing this comes from Neal @ Tech Help Notes, as recommended by Dr. Middleton in one of her lectures. A lower standard error generally indicates a better model, because it indicates that there is less variance between our model and the actual data points. The initial multiple regression model in D1 had a residual standard error of 1.113. The residual standard error for this final model is 0.016. This is a smaller standard error, indicating that this a better model.

D3: Reduced Multiple Regression Model

The variables for Vitamin D levels and doctor visits were eliminated due to multicolinearity, and the columns for number of children, age, income, gender, and admission for observation were all eliminated due to poor p-values. The columns with poor p-values also could be noted in the initial multiple regression model as having the smallest coefficients, as well, being significantly smaller than the remaining variables. The variables remaining in the model and having the most impact on the dependent (y) variable of days hospitalized are: - initial admission - emergency - complication risk - arthritis - diabetes - back pain - average daily charge amount

These variables are seen in the final reduced multiple regression model, which has a reduced residual standard error compared to the initial model:

[ ]:
```python
# All p-values for independent variables are < 0.05, this is the final reduced␣
↪model
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["initial_admit_emerg", "comp_risk_low", "comp_risk_medium",␣
↪"arthritis", "diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)
```

```
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 7.976e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:05   Log-Likelihood:                 27373.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9992   BIC:                        -5.467e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
initial_admit_emerg    -0.0884      0.000   -280.292      0.000      -0.089
-0.088
comp_risk_low           0.0716      0.000    164.548      0.000       0.071
0.072
comp_risk_medium        0.0709      0.000    197.823      0.000       0.070
0.072
arthritis              -0.0128      0.000    -39.038      0.000      -0.013
-0.012
diabetes               -0.0129      0.000    -36.778      0.000      -0.014
-0.012
back_pain              -0.0150      0.000    -46.995      0.000      -0.016
-0.014
avg_daily_charge        1.2428      0.001   2361.509      0.000       1.242
1.244
const                  -0.0959      0.000   -217.782      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      194.884   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.109
Skew:                          -0.248   Prob(JB):                     2.31e-36
Kurtosis:                       2.616   Cond. No.                         6.09
==============================================================================

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

E1: Analysis of Multiple Regression Models

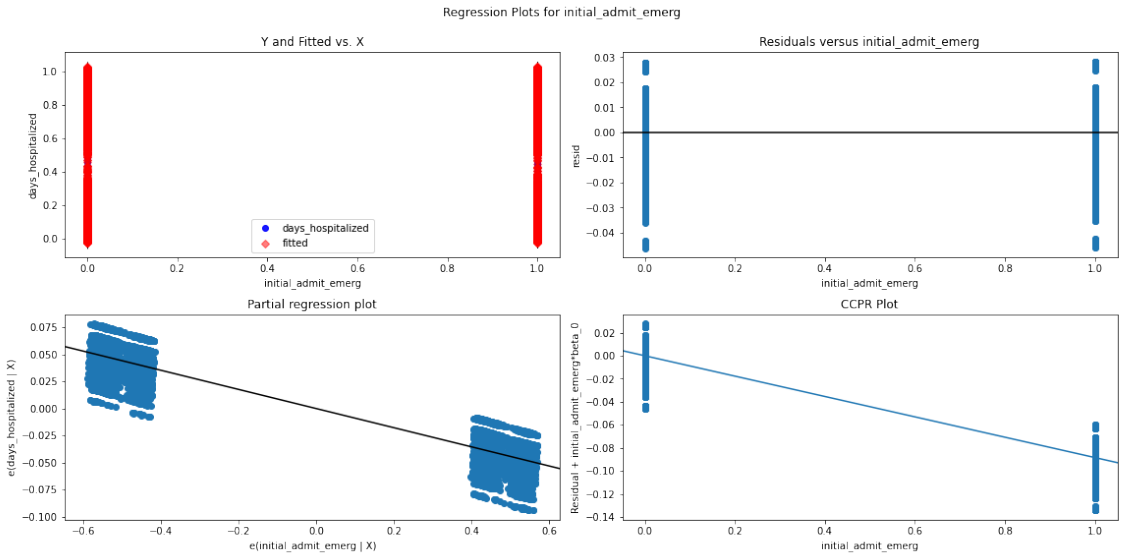
The initial multiple regression model had a lot of variables in it, not all of which were particularly important to the model itself. Two variables were removed due to multicolinearity concerns, vit_d_levels and dr_visits. After those two were eliminated, other variables were removed from the initial model through a process of Backwards Stepwise Elimination based on the p-value of each remaining variable. The p-value of a variable indicates if it is statistically significant or not, with lower values being more significant. If a variable isn't statistically significant, then it doesn't need to be kept in the model. Eliminating the variable with the highest (worst) p-value can change how other variables interact, so this has to be done one at a time, rerunning the model after each variable is eliminated. This was done until every variable remaining had a p-value less than 0.05, indicating that the variable was statistically significant.

As mentioned above, both the initial and the reduced multiple regression models have p-values indicating that they are statistically significant, as well as having very high r-squared values. As a result, comparison of the two needed to use an alternative metric to demonstrate the model's improvement (or lackthereof). This was done using the residual standard error.

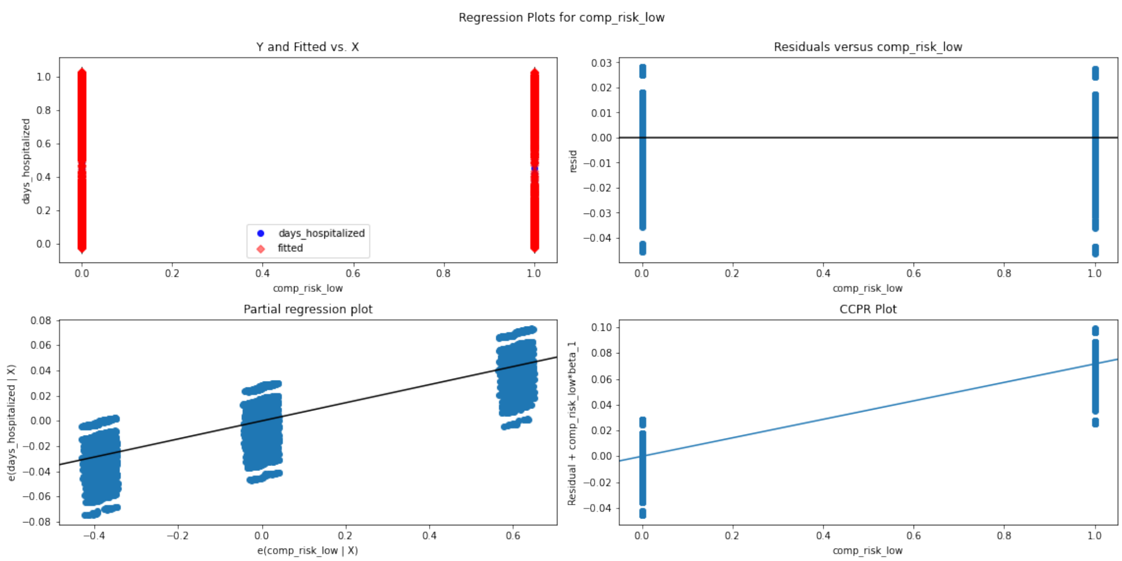
When a model is created, it is rarely an absolutely perfect, usually having some amount of difference between the actual data and the line of best fit. These differences are called the model's residuals, one way to assess it is to measure the residuals, the differences between the actual data and the line of best fit. A lower standard error of these residuals is generally indicative of a better model, as it indicates less variance in the magnitude of the residuals and this error between their actual placement and their projection. The initial multiple regression model seen in section D1 had a residual standard error of 1.113, while the reduced model in section D3 had a residual standard error of just 0.016. This indicates that the reduced model is a better overall model for projecting days_hospitalized than the initial model was.

Residual plots for the model are below. I provided residual plots for each explanatory variable, as the rubric didn't specify besides requiring "a" residual plot". Code for this was generated with assistance from the GeeksForGeeks page on generating residual plots in python.

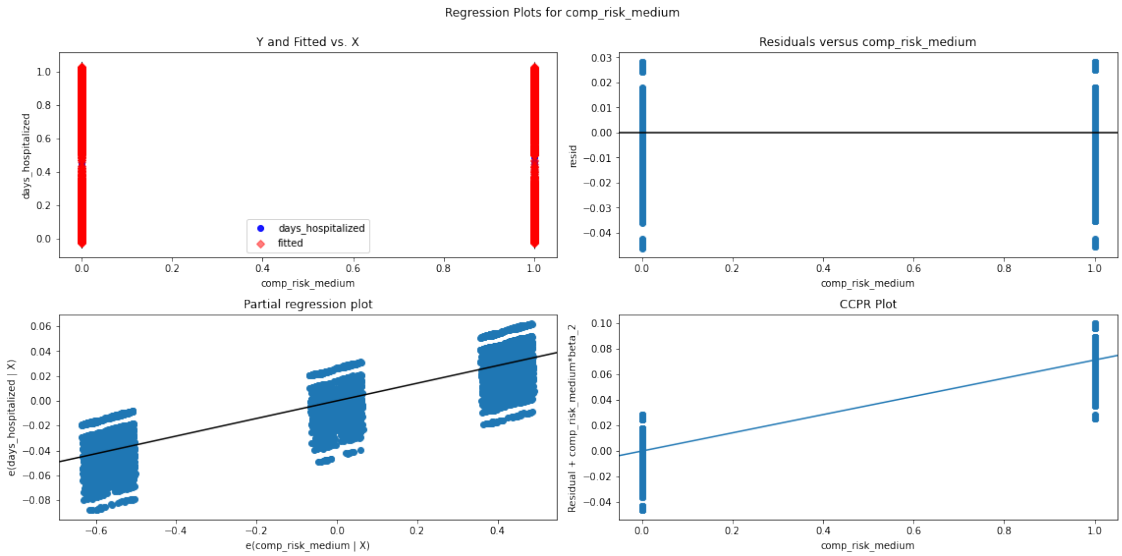
[ ]:
```
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'initial_admit_emerg', fig=fig);
```



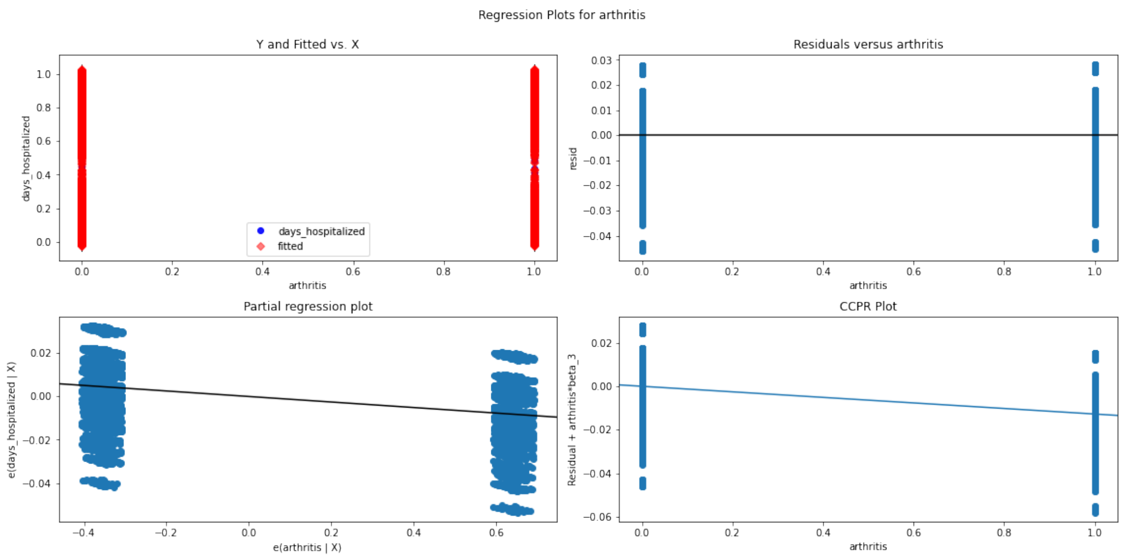
[ ]:
```
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'comp_risk_low', fig=fig);
```



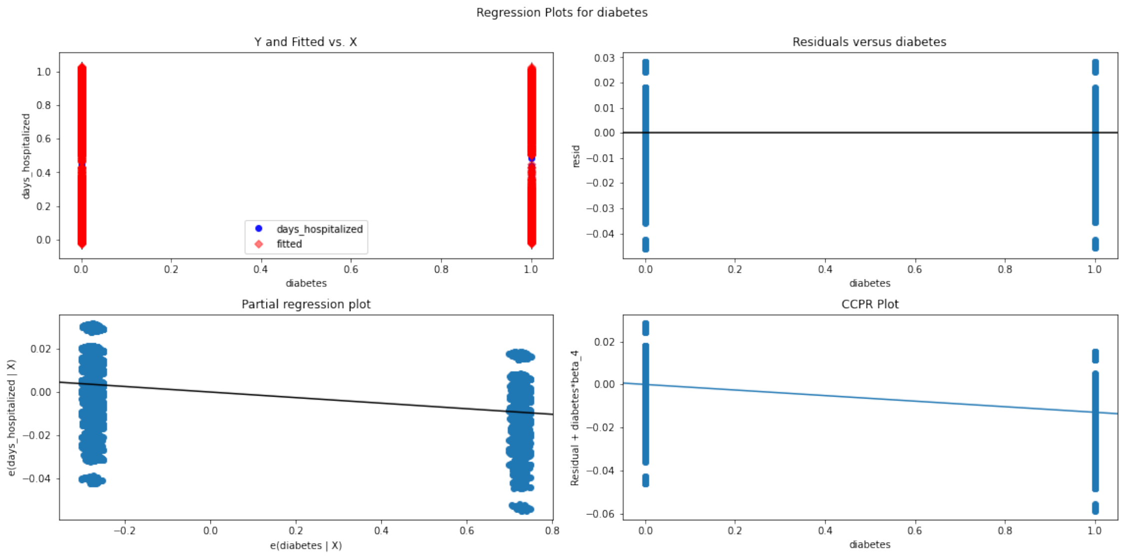
[ ]:
```
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'comp_risk_medium', fig=fig);
```



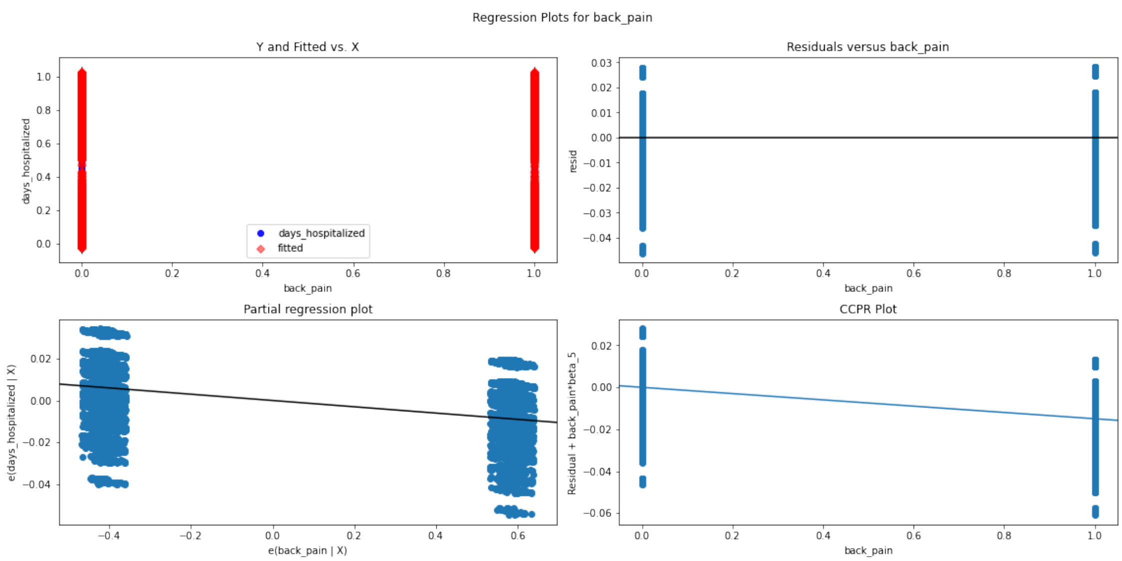

```
[ ]: fig = plt.figure(figsize = [16,8])
     sm.graphics.plot_regress_exog(results, 'arthritis', fig=fig);
```



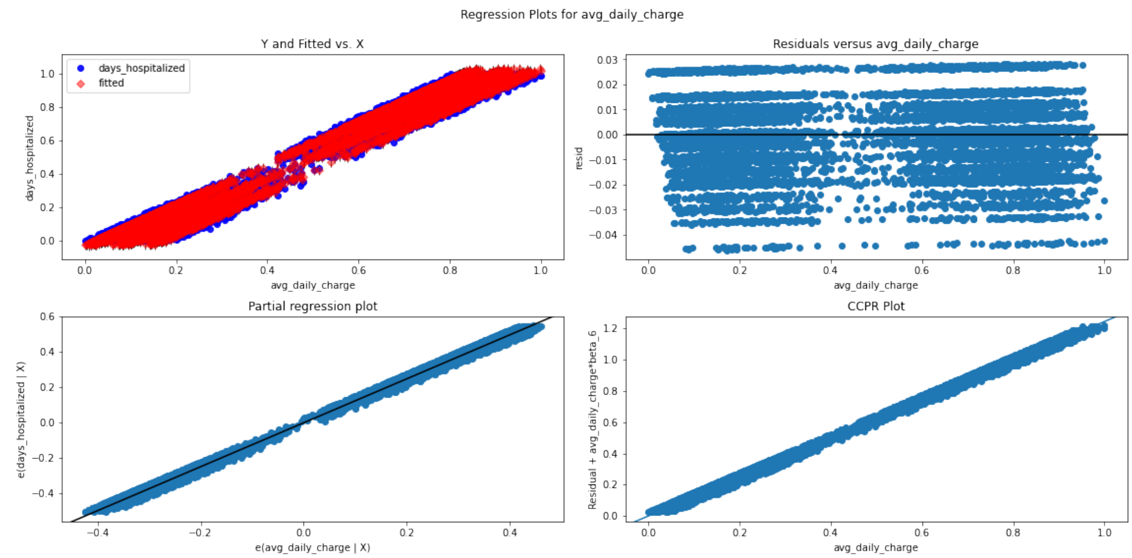

```
[ ]: fig = plt.figure(figsize = [16,8])
     sm.graphics.plot_regress_exog(results, 'diabetes', fig=fig);
```



[ ]:
```python
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'back_pain', fig=fig);
```



[ ]:
```python
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'avg_daily_charge', fig=fig);
```



These residual plots seem to indicate that the residuals are not homoscedastic, or at least that they are not ideally homoscedastic. In most of the above plots, the residuals are not centered upon 0 or the line of best fit. Instead, most of them occur below this point. This can be seen somewhat with the categorical variables, but it is best observed with the avg_daily_charge variable. With that residual plot, we can see that the residuals range from approaching +0.03 to exceeding -0.04, with many of them occurring at or above -0.03 in magnitude. While this does not appear to be completely heteroscedastic, it does indicate that there is a skew here and that we may not be satisfying the underlying assumptions necessary for multiple regression.

E2: Model Outputs

This part of the rubric is confusing, as I've literally been asked to provide these outputs in sections D and E1, and again in E3. I will provide these outputs again in section E3, where I compile all of the required code *and* the outputs of each.

E3: Model Code

[ ]:
```
# Initial model for Multiple Linear Regression

# Set dependent variable
y = regress_df.days_hospitalized
# Set multiple independent variables
X = regress_df[["num_children", "age", "income", "gender_male", "gender_nonbinary", "vit_d_level", "dr_visits", "initial_admit_observ", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 3.986e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:20   Log-Likelihood:                -15249.
No. Observations:               10000   AIC:                         3.053e+04
Df Residuals:                    9985   BIC:                         3.064e+04
Df Model:                          14
Covariance Type:            nonrobust
================================================================================
========
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children            0.0012      0.005      0.229      0.819      -0.009
0.011
age                    -0.0003      0.001     -0.561      0.575      -0.001
0.001
income              -3.601e-07    3.9e-07     -0.922      0.356   -1.13e-06
4.05e-07
gender_male            -0.0006      0.011     -0.057      0.955      -0.022
0.021
gender_nonbinary       -0.0006      0.011     -0.057      0.955      -0.022
0.021
vit_d_level             0.0029      0.006      0.525      0.600      -0.008
0.014
dr_visits               0.0096      0.011      0.899      0.369      -0.011
0.030
initial_admit_observ    0.0202      0.032      0.637      0.524      -0.042
0.082
initial_admit_emerg    -6.2641      0.027   -229.353      0.000      -6.318
-6.211
comp_risk_low           5.0828      0.031    164.481      0.000       5.022
5.143
comp_risk_medium        5.0335      0.025    197.661      0.000       4.984
5.083
arthritis              -0.9072      0.023    -39.028      0.000      -0.953
-0.862
diabetes               -0.9179      0.025    -36.750      0.000      -0.967
-0.869
back_pain              -1.0633      0.023    -46.947      0.000      -1.108
-1.019
avg_daily_charge        0.0122   5.16e-06   2359.774      0.000       0.012
0.012
const                 -29.4951      0.124   -237.141      0.000     -29.739
```

```
-29.251
==============================================================================
Omnibus:                      195.513   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              165.223
Skew:                          -0.250   Prob(JB):                     1.33e-36
Kurtosis:                       2.618   Cond. No.                     3.47e+20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 2.06e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[ ]:
```
results.resid.std(ddof=X.shape[1])
```

[ ]:
```
1.1126862546385587
```

[ ]:
```
# Warning exists for multicolinearity - check for VIF to see if variables␣
↪should be eliminated due to high multicolinearity
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "vit_d_level", "dr_visits", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]]

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                 feature        VIF
0           num_children   1.932741
1                    age   7.361227
2                 income   2.963630
3            gender_male        inf
4       gender_nonbinary        inf
5            vit_d_level  29.950450
6              dr_visits  19.835725
7   initial_admit_observ   1.950991
8    initial_admit_emerg   3.014566
9          comp_risk_low   1.618548
10      comp_risk_medium   2.324442
11            arthritis   1.555091
12             diabetes   1.373587
13            back_pain   1.697394
```

```
14      avg_daily_charge   6.760867

C:\Users\hasek\anaconda3\lib\site-
packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by
zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

[ ]:
```
# Eliminated vit_d_level (VIF = 29.95), rerunning analysis to see if any VIF␣
↪still above 10
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "dr_visits", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]]

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                 feature        VIF
0            num_children   1.909810
1                     age   6.650950
2                  income   2.888846
3             gender_male        inf
4        gender_nonbinary        inf
5               dr_visits  12.338329
6    initial_admit_observ   1.904371
7     initial_admit_emerg   2.919809
8           comp_risk_low   1.587692
9        comp_risk_medium   2.244750
10              arthritis   1.547064
11               diabetes   1.370846
12              back_pain   1.688951
13       avg_daily_charge   6.218787

C:\Users\hasek\anaconda3\lib\site-
packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by
zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

[ ]:
```
# Eliminated dr_visits (VIF = 12.34), rerunning analysis to see if any VIF␣
↪still above 10
X = regress_df[["num_children", "age", "income", "gender_male",␣
↪"gender_nonbinary", "initial_admit_observ", "initial_admit_emerg",␣
↪"comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
↪"avg_daily_charge"]]
```

```
vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                feature       VIF
0          num_children  1.880773
1                   age  5.568718
2                income  2.740994
3           gender_male       inf
4      gender_nonbinary       inf
5   initial_admit_observ  1.810973
6    initial_admit_emerg  2.769873
7         comp_risk_low  1.548161
8      comp_risk_medium  2.148653
9             arthritis  1.535005
10             diabetes  1.358931
11            back_pain  1.670107
12     avg_daily_charge  5.435098
```

```
C:\Users\hasek\anaconda3\lib\site-
packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by
zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
```

[ ]:
```
reg_df_minmax = pd.DataFrame(preprocessing.MinMaxScaler().
 ↪fit_transform(regress_df), columns=regress_df.columns)
reg_df_minmax
```

[ ]:
```
      num_children       age    income  gender_male  gender_nonbinary  \
0              0.1  0.492958  0.417301          1.0               1.0
1              0.3  0.464789  0.225264          0.0               0.0
2              0.3  0.492958  0.068645          0.0               0.0
3              0.0  0.845070  0.191154          1.0               1.0
4              0.1  0.056338  0.005094          0.0               0.0
…              …        …        …            …                 …
9995           0.2  0.098592  0.221217          1.0               1.0
9996           0.4  0.971831  0.071605          1.0               1.0
9997           0.3  0.380282  0.317550          0.0               0.0
9998           0.3  0.352113  0.142678          1.0               1.0
9999           0.8  0.732394  0.301929          0.0               0.0

      vit_d_level  dr_visits  initial_admit_observ  initial_admit_emerg  \
0        0.562756      0.625                   0.0                  1.0
```

```
1         0.550632      0.375                    0.0                    1.0
2         0.497410      0.375                    0.0                    0.0
3         0.408150      0.375                    0.0                    0.0
4         0.460128      0.500                    0.0                    0.0
…             …        …                   …                   …
9995      0.432505      0.375                    0.0                    1.0
9996      0.504615      0.500                    0.0                    0.0
9997      0.441440      0.375                    0.0                    0.0
9998      0.609113      0.500                    0.0                    1.0
9999      0.517371      0.500                    1.0                    0.0

      comp_risk_low  comp_risk_medium  arthritis  diabetes  back_pain  \
0               0.0               1.0        1.0       1.0        1.0
1               0.0               0.0        0.0       0.0        0.0
2               0.0               1.0        0.0       1.0        0.0
3               0.0               1.0        1.0       0.0        0.0
4               1.0               0.0        0.0       0.0        0.0
…             …                …         …        …        …
9995            0.0               1.0        0.0       0.0        0.0
9996            0.0               1.0        1.0       1.0        0.0
9997            0.0               0.0        0.0       0.0        0.0
9998            0.0               1.0        0.0       0.0        1.0
9999            1.0               0.0        1.0       0.0        0.0

      days_hospitalized  avg_daily_charge
0              0.135022          0.246933
1              0.199037          0.311343
2              0.053117          0.068475
3              0.010044          0.026168
4              0.003562          0.024130
…                  …                …
9995           0.712308          0.678314
9996           0.953321          0.801304
9997           0.974256          0.875146
9998           0.878492          0.787882
9999           0.984067          0.821444

[10000 rows x 16 columns]
```

[ ]:
```
# Reduce model by eliminating dr_visits and vit_d_levels due to
↪multicolinearity issues
# Continue reducing model by backward elimination, removing the highest p-value
↪(alpha = 0.05) and re-modelling
# will eventually be left with only the most important variables

# BACKWARD ELIMINATION # 1: Seek highest p-value above 0.05 (eliminated
↪dr_visits, vit_d_levels due to multicolinearity)
```

```python
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["num_children", "age", "income", "gender_male",␣
 ↪"gender_nonbinary", "initial_admit_observ", "initial_admit_emerg",␣
 ↪"comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
 ↪"avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 4.651e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.472e+04
Df Residuals:                    9987   BIC:                        -5.463e+04
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children             0.0002      0.001      0.232      0.817      -0.001
0.002
age                     -0.0003      0.001     -0.549      0.583      -0.001
0.001
income                  -0.0010      0.001     -0.917      0.359      -0.003
0.001
gender_male          -1.072e-05      0.000     -0.068      0.946      -0.000
0.000
gender_nonbinary     -1.072e-05      0.000     -0.068      0.946      -0.000
0.000
initial_admit_observ     0.0003      0.000      0.658      0.510      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.429      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.489      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.684      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.030      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.770      0.000      -0.014
```

```
-0.012
back_pain              -0.0150      0.000    -46.950      0.000      -0.016
-0.014
avg_daily_charge        1.2428      0.001   2359.944      0.000       1.242
1.244
const                  -0.0957      0.001   -149.616      0.000      -0.097
-0.094
==============================================================================
Omnibus:                      195.218   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.570
Skew:                          -0.249   Prob(JB):                     1.84e-36
Kurtosis:                       2.617   Cond. No.                     1.75e+17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 1.02e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[ ]:
```python
# BACKWARD ELIMINATION # 2: Seek highest p-value above 0.05 (eliminated␣
↪gender_male, p-value of 0.946)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["num_children", "age", "income", "gender_nonbinary",␣
↪"initial_admit_observ", "initial_admit_emerg", "comp_risk_low",␣
↪"comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
↪"avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 4.651e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.472e+04
Df Residuals:                    9987   BIC:                        -5.463e+04
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
--------
num_children             0.0002      0.001      0.232      0.817      -0.001
0.002
age                     -0.0003      0.001     -0.549      0.583      -0.001
0.001
income                  -0.0010      0.001     -0.917      0.359      -0.003
0.001
gender_nonbinary     -2.143e-05      0.000     -0.068      0.946      -0.001
0.001
initial_admit_observ     0.0003      0.000      0.658      0.510      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.429      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.489      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.684      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.030      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.770      0.000      -0.014
-0.012
back_pain               -0.0150      0.000    -46.950      0.000      -0.016
-0.014
avg_daily_charge         1.2428      0.001   2359.944      0.000       1.242
1.244
const                   -0.0957      0.001   -149.616      0.000      -0.097
-0.094
==============================================================================
Omnibus:                      195.218   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.570
Skew:                          -0.249   Prob(JB):                     1.84e-36
Kurtosis:                       2.617   Cond. No.                         12.6
==============================================================================

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

[ ]:

```
# BACKWARD ELIMINATION # 3: Seek highest p-value above 0.05 (eliminated␣
 ↪gender_nonbinary, p-value of 0.946)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["num_children", "age", "income", "initial_admit_observ",␣
 ↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
 ↪"diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
```

```python
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 5.074e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.472e+04
Df Residuals:                    9988   BIC:                        -5.464e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
num_children             0.0002      0.001      0.232      0.816      -0.001
0.002
age                     -0.0003      0.001     -0.548      0.584      -0.001
0.001
income                  -0.0010      0.001     -0.917      0.359      -0.003
0.001
initial_admit_observ     0.0003      0.000      0.660      0.509      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.504      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.497      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.699      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.033      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.771      0.000      -0.014
-0.012
back_pain               -0.0150      0.000    -46.957      0.000      -0.016
-0.014
avg_daily_charge         1.2428      0.001   2360.121      0.000       1.242
1.244
const                   -0.0957      0.001   -154.535      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.200   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.541
Skew:                          -0.249   Prob(JB):                     1.86e-36
Kurtosis:                       2.617   Cond. No.                         12.0
```

```
==============================================================================

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

[ ]:

```
# BACKWARD ELIMINATION # 4: Seek highest p-value above 0.05 (eliminated␣
↪num_children, p-value of 0.816)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["age", "income", "initial_admit_observ",␣
↪"initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis",␣
↪"diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 5.582e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9989   BIC:                        -5.465e+04
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
age                     -0.0003      0.001     -0.546      0.585      -0.001
0.001
income                  -0.0010      0.001     -0.916      0.360      -0.003
0.001
initial_admit_observ     0.0003      0.000      0.660      0.509      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.516      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.505      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.710      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.034      0.000      -0.013
-0.012
```

```
diabetes                   -0.0129      0.000    -36.777      0.000      -0.014
-0.012
back_pain                  -0.0150      0.000    -46.963      0.000      -0.016
-0.014
avg_daily_charge            1.2428      0.001   2360.835      0.000       1.242
1.244
const                      -0.0957      0.001   -158.610      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.107   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.502
Skew:                          -0.249   Prob(JB):                     1.90e-36
Kurtosis:                       2.617   Cond. No.                         11.9
==============================================================================

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

[ ]:
```python
# BACKWARD ELIMINATION # 5: Seek highest p-value above 0.05 (eliminated age, p-value of 0.585)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["income", "initial_admit_observ", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "arthritis", "diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 6.203e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27374.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9990   BIC:                        -5.466e+04
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
income                  -0.0010      0.001     -0.909      0.363      -0.003
```

```
0.001
initial_admit_observ     0.0003      0.000      0.667      0.505      -0.001
0.001
initial_admit_emerg     -0.0882      0.000   -229.536      0.000      -0.089
-0.087
comp_risk_low            0.0716      0.000    164.512      0.000       0.071
0.072
comp_risk_medium         0.0709      0.000    197.722      0.000       0.070
0.072
arthritis               -0.0128      0.000    -39.040      0.000      -0.013
-0.012
diabetes                -0.0129      0.000    -36.780      0.000      -0.014
-0.012
back_pain               -0.0150      0.000    -46.986      0.000      -0.016
-0.014
avg_daily_charge         1.2428      0.001   2361.202      0.000       1.242
1.244
const                   -0.0958      0.001   -177.580      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.297   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.661
Skew:                          -0.249   Prob(JB):                     1.75e-36
Kurtosis:                       2.617   Cond. No.                         11.3
==============================================================================

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

[ ]:
```python
# BACKWARD ELIMINATION # 6: Seek highest p-value above 0.05 (eliminated␣
↪initial_admit_observ, p-value of 0.505)
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["income", "initial_admit_emerg", "comp_risk_low",␣
↪"comp_risk_medium", "arthritis", "diabetes", "back_pain",␣
↪"avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 6.979e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27374.
```

```
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9991   BIC:                        -5.466e+04
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
income                 -0.0010      0.001     -0.902      0.367      -0.003
0.001
initial_admit_emerg    -0.0884      0.000   -280.249      0.000      -0.089
-0.088
comp_risk_low           0.0716      0.000    164.548      0.000       0.071
0.072
comp_risk_medium        0.0709      0.000    197.812      0.000       0.070
0.072
arthritis              -0.0128      0.000    -39.042      0.000      -0.013
-0.012
diabetes               -0.0129      0.000    -36.785      0.000      -0.014
-0.012
back_pain              -0.0150      0.000    -46.983      0.000      -0.016
-0.014
avg_daily_charge        1.2428      0.001   2361.289      0.000       1.242
1.244
const                  -0.0957      0.000   -192.771      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      195.261   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.491
Skew:                          -0.249   Prob(JB):                     1.91e-36
Kurtosis:                       2.616   Cond. No.                         11.1
==============================================================================

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

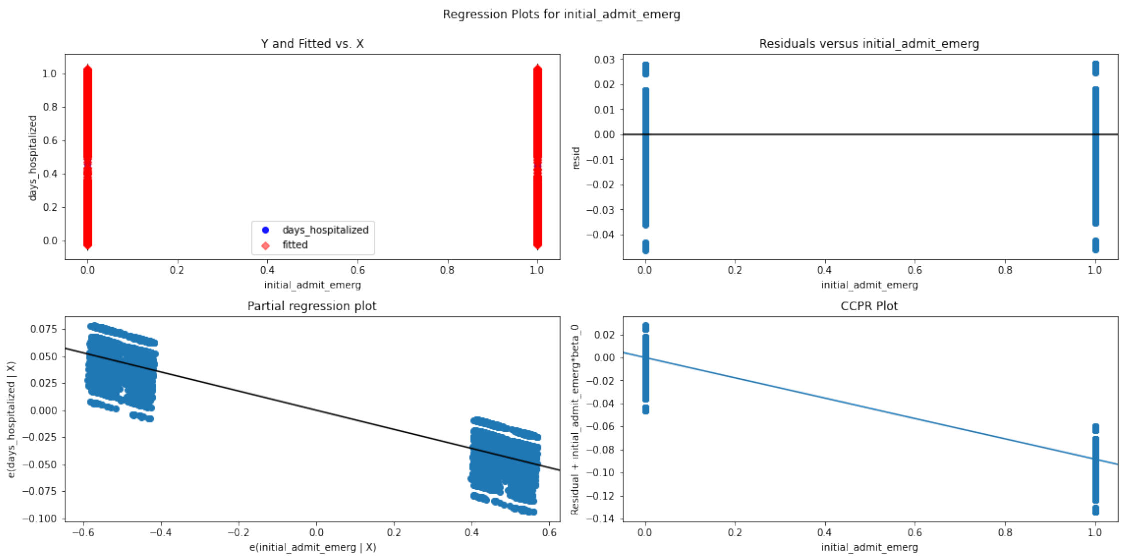
[ ]:
```python
# BACKWARD ELIMINATION # 7: Eliminated income, this is the final model
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["initial_admit_emerg", "comp_risk_low", "comp_risk_medium",
 ↪"arthritis", "diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 7.976e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:21   Log-Likelihood:                 27373.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9992   BIC:                        -5.467e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
=======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
initial_admit_emerg   -0.0884      0.000   -280.292      0.000      -0.089
-0.088
comp_risk_low          0.0716      0.000    164.548      0.000       0.071
0.072
comp_risk_medium       0.0709      0.000    197.823      0.000       0.070
0.072
arthritis             -0.0128      0.000    -39.038      0.000      -0.013
-0.012
diabetes              -0.0129      0.000    -36.778      0.000      -0.014
-0.012
back_pain             -0.0150      0.000    -46.995      0.000      -0.016
-0.014
avg_daily_charge       1.2428      0.001   2361.509      0.000       1.242
1.244
const                 -0.0959      0.000   -217.782      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      194.884   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.109
Skew:                          -0.248   Prob(JB):                     2.31e-36
Kurtosis:                       2.616   Cond. No.                         6.09
==============================================================================

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

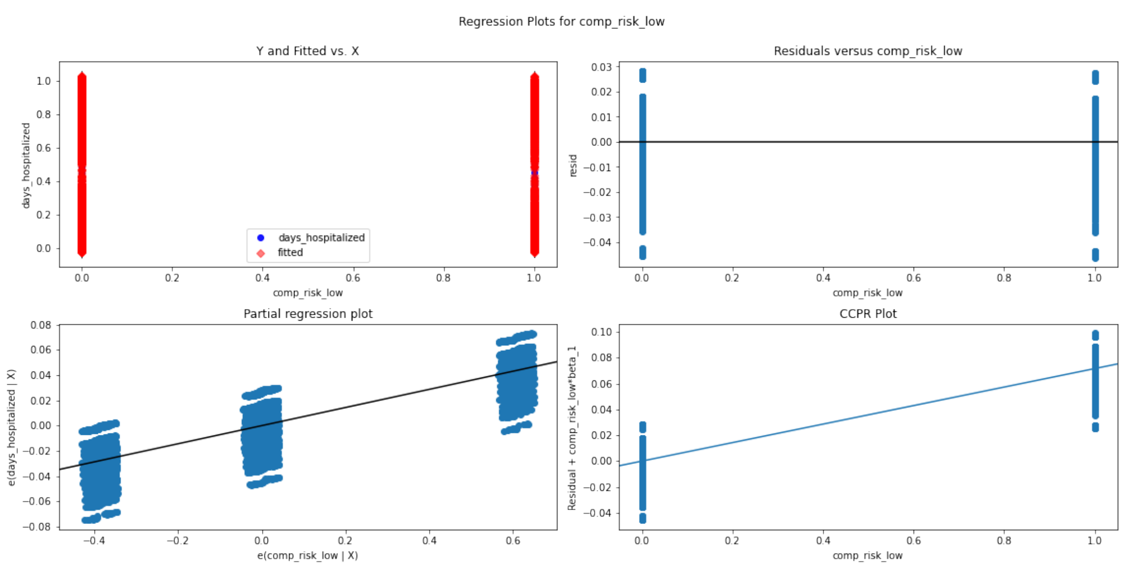
[ ]:
```
results.resid.std(ddof=X.shape[1])
```

[ ]: 0.01567200935795125

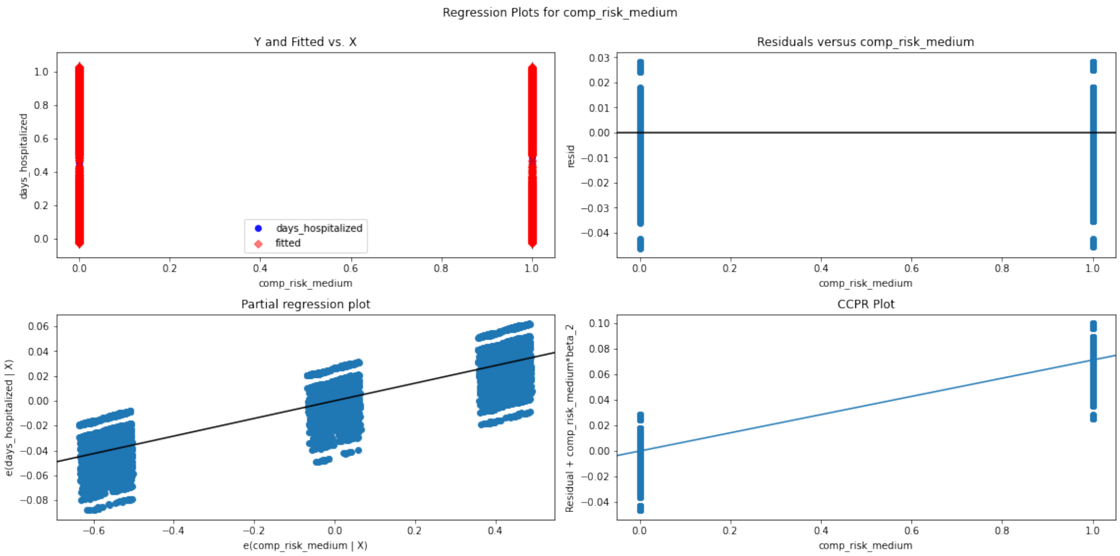
[ ]:
```python
# Residual plot for initial_admit_emerg
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'initial_admit_emerg', fig=fig);
```



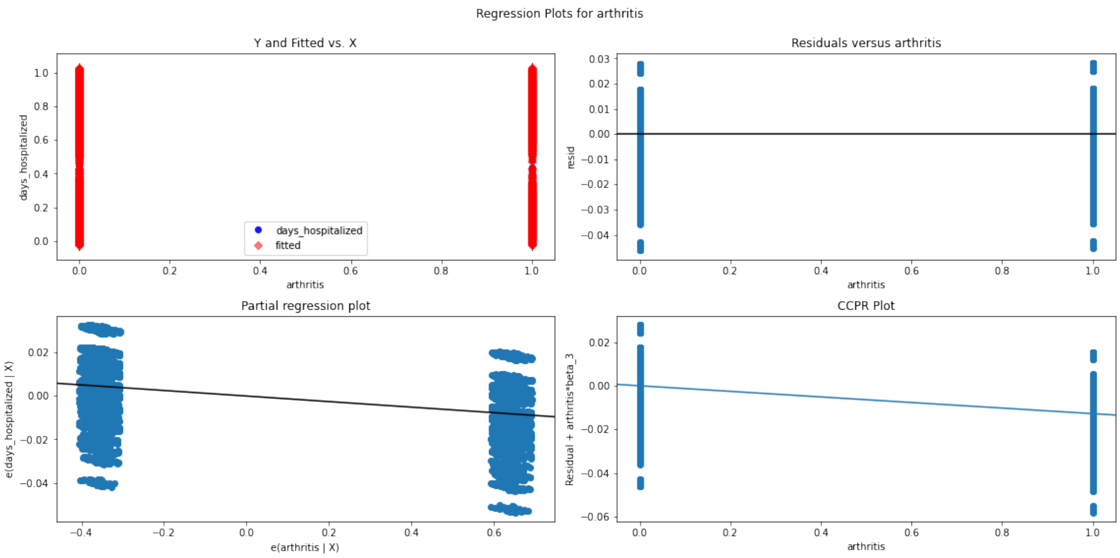
[ ]:
```python
# Residual plot for comp_risk_low
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'comp_risk_low', fig=fig);
```



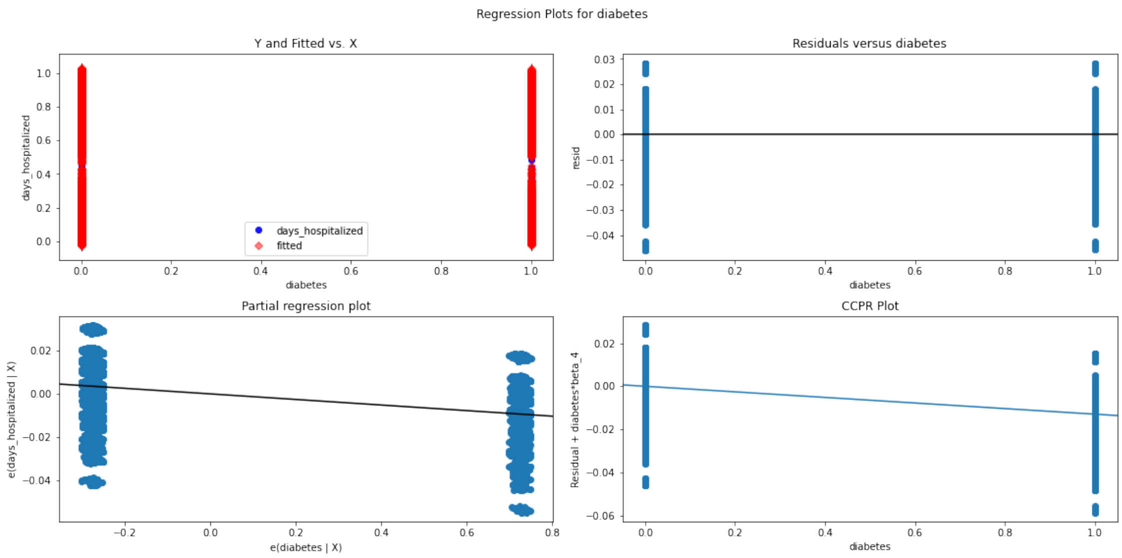
[ ]:
```python
# Residual plot for comp_risk_medium
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'comp_risk_medium', fig=fig);
```



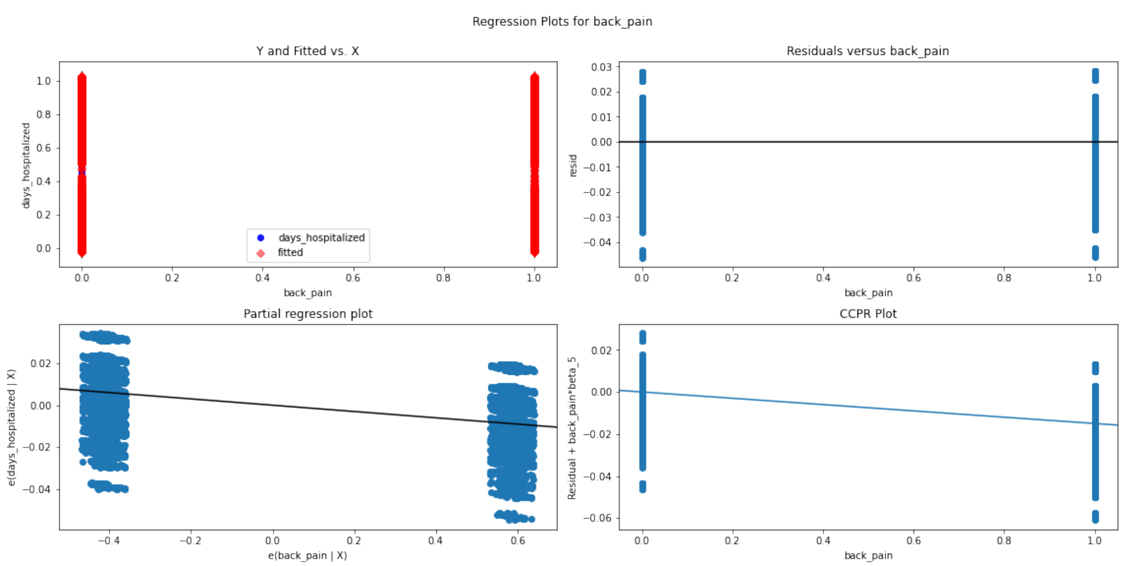
[ ]:
```python
# Residual plot for arthritis
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'arthritis', fig=fig);
```



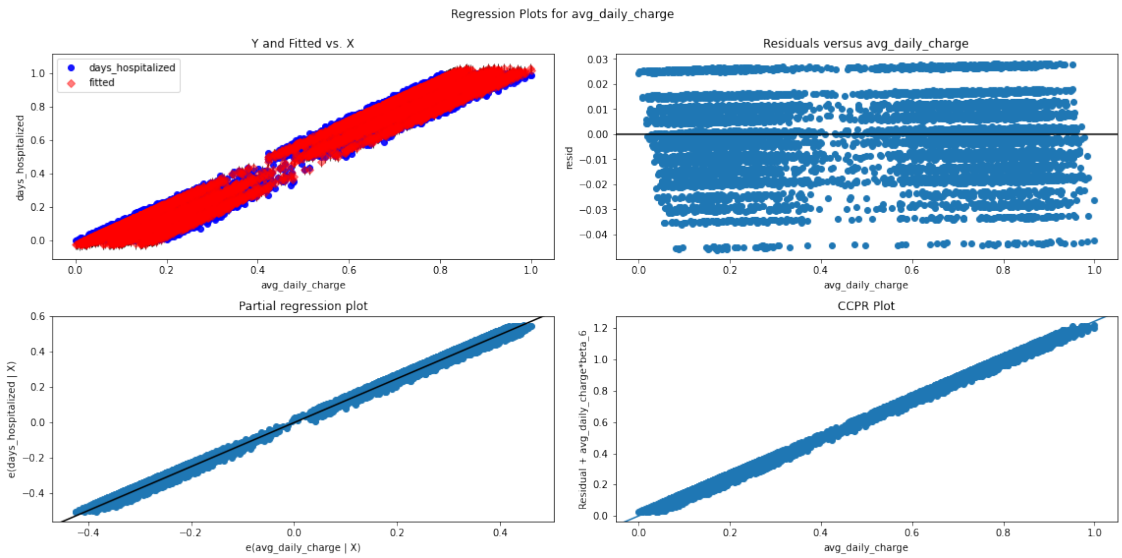
```
[ ]: # Residual plot for diabetes
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'diabetes', fig=fig);
```




```
[ ]: # Residual plot for back_pain
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'back_pain', fig=fig);
```



[ ]:
```
# Residual plot for avg_daily_charge
fig = plt.figure(figsize = [16,8])
sm.graphics.plot_regress_exog(results, 'avg_daily_charge', fig=fig);
```



[ ]:
```
y = reg_df_minmax.days_hospitalized
X = reg_df_minmax[["initial_admit_emerg", "comp_risk_low", "comp_risk_medium",␣
↪"arthritis", "diabetes", "back_pain", "avg_daily_charge"]].assign(const=1)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      days_hospitalized   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 7.976e+05
Date:                Wed, 16 Nov 2022   Prob (F-statistic):               0.00
Time:                        21:02:38   Log-Likelihood:                 27373.
No. Observations:               10000   AIC:                        -5.473e+04
Df Residuals:                    9992   BIC:                        -5.467e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
```

```
initial_admit_emerg    -0.0884      0.000   -280.292      0.000      -0.089
-0.088
comp_risk_low           0.0716      0.000    164.548      0.000       0.071
0.072
comp_risk_medium        0.0709      0.000    197.823      0.000       0.070
0.072
arthritis              -0.0128      0.000    -39.038      0.000      -0.013
-0.012
diabetes               -0.0129      0.000    -36.778      0.000      -0.014
-0.012
back_pain              -0.0150      0.000    -46.995      0.000      -0.016
-0.014
avg_daily_charge        1.2428      0.001   2361.509      0.000       1.242
1.244
const                  -0.0959      0.000   -217.782      0.000      -0.097
-0.095
==============================================================================
Omnibus:                      194.884   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              164.109
Skew:                          -0.248   Prob(JB):                     2.31e-36
Kurtosis:                       2.616   Cond. No.                         6.09
==============================================================================

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

F1: Results of Data Analysis

The multiple regression analysis yields the following equation for the reduced model:

$$\hat{y} = -0.0959+1.2428(mean\ daily\ charge)-0.0150(back\ pain)-0.0129(diabetes)-0.0128(arthritis)+0.0709(mediu$$

This can be used to conclude the following about each explanatory variable, using the format provided in the WGU Courseware Resources: - Keeping all things constant, one unit increase in average daily charge is associated with a 1.2428 *increase* in days hospitalized - Keeping all things constant, patients with chronic back pain spend 1.5% *fewer* days hospitalized. - Keeping all things constant, patients with diabetes spend 1.29% *fewer* days hospitalized - Keeping all things constant, patients with arthritis spend 1.28% *fewer* days hospitalized - Keeping all things constant, patients with low complication risk spend 7.09% *more* days hospitalized - Keeping all things constant, patients with medium complication risk spend 7.16% *more* days hospitalized - Keeping all things constant, patients admitted for an emergency spend 8.84% *fewer* days hospitalized

The models significance can be considered from both a statistical and a practical perspective. The model does appear to be statistically significant, based upon the probability of its f-statistic being 0.00. This is less than 0.05, indicating that it is statistically significant and not based on random chance. The deviance from perfect homoscedasticity could indicate some concerns with the accuracy of the model, but if we accept that it is "in the ballpark", then I'm actually not concerned about this because I don't think the model is practically significant. The main reason that I don't find this model to have much in the way of practical significance is that the variables that it identified as significant are generally out of the control of the hospital system.

Variables such as a patient having back pain or diabetes aren't anything that the hospital can really control, as these are entirely determined by the patient. A hospital, unlike some other businesses, doesn't really get to choose who it provides services to. The nature of a hospital is that someone walks in the door and they get treatment, regardless of what problems they have - and *that's a good thing*. Moreover, the variables that are "decided" by the patient (back pain, diabetes, arthritis, and whether they come in for an emergency) are all actually negative, decreasing the length of hospitalization. The variables that increase hospitalization length are average daily charge and omplication risk. The complication risk's impact is intuitive, because obviously more significant problems with higher complication risks are simply going to have complications more often - its self-evident and the process of assessment and triage is necessary to managing care, especially in the emergency room.

As for daily charge, this is maybe the only thing that is really within the hospital's control, but its not meaningful in any sense. Average daily charge has the largest input on how long someone is hospitalized. This makes sense in some regard, because the longer someone is hospitalized, the more that their hospital bill ends up being. However, the amount billed doesn't actually relate to how long a person is hospitalized. It is instead a result of many things, most primarily the length of hospitalization and the treatments received. This means that length of hospitalization is actually influencing the billing, rather than the billing influencing the length of hospitalization. This can be demonstrated with a very easy thought experiment. If a patient stayed in the hospital but were billed $ for their stay and treatment, it would not reduce their stay. Conversely, if a patient stayed in the hospital but were afterwards sent a final bill for 20 times the normal rate for their stay and treatment, this would not lengthen the patient's hospital stay. For this reason, I actually considered excluding this variable from my analysis in the first place. However, given few continuous variables and a rubric requirement to include them, I ended up keeping this in the analysis.

Regarding the limitations of this analysis, several limitations occurred to me in preparing the data and generating the analysis. This is not an exhaustive list of concerns, especially because the analysis itself lacks practical significance, but here are several that I noted throughout the project, in addition to those highlighted in the above discussion on practical significance:

- The hospitalization data only includes patients hospitalized for at least 24 hours This data ignores any patient hospitalized for less than 1 full day. Avoiding hospitalization in the first place is obviously the first goal of patients and hospitals, and from a practical perspective, there may be a difference between a patient occupying a temporary place in an emergency room versus being formally admitted and roomed. It is possible that this creates different incentives that might not be seen in this data, or that the relationships seen in this analysis may not necessarily apply (or may be stronger or weaker) within the first 24 hours of someone's hospital care.
- It is unclear if this data is inclusive only of patients whose admission ended with a discharge to the street, or if this includes patients who may have died while hospitalized While grim to consider, it is a fact that many patients die in hospitals, for a variety of reasons. Given that the original basis of this dataset is that it is about readmission to the hospital, patients who died may not be included because they're obviously not going to be readmitted. There is also a very cynical reason to include them in these statistics, because these patients would

actually make readmission numbers look better than they are, by including patients who by definition would never be readmitted. Patients dying in hospital care represents a failure on the part of medical staff in at least some circumstances, and examination of any dataset while omitting one's failures is going to necessarily create a biased conclusion.

- Patient income should not be collected at all, and if it is collected, should be presented in categorical "buckets". As stated above in the process of preparing the data, the income figures in this dataset are strange, being calculated down to the 6th decimal place. Even being rounded to the single dollar is not a realistic way that anyone discusses their annual income in the United States. This gives me serious concerns about the accuracy of the data and where it is coming from, as it seems to be generated programmatically, rather than from the patient themselves. Income should have no bearing on the way a patient is treated nor the medical decisions made regarding their care, so it has no business being collected. Even if we assume that for some reason it was of interest, there is no practical difference between one patient who makes $80,000 in annual income, and an identical patient who makes one dollar more. This would be best represented as a categorical variable representing various income ranges, rather than a continuous variable offering the illusion of precision that has no value in the real world.
- The dataset does not contain any patients younger than age 18 The omission of any patients below the age of 18 is potentially problematic, in that trends or relationships seen in this analysis may not necessarily hold true for children. Creating policy or formalizing "best practices" based on this data might be problematic if those policies or practices are applied to minors, based on an analysis that omitted minors. While this might not be a problem with a teenaged minor who is mostly grown and approaching adulthood, this could be especially problematic with younger children or especially infants.

F2: Recommended Action

Given that I've concluded that my model doesn't have much in the way of practical significance, there's not a whole lot that I can recommend based upon this analysis. I would have two primary recommendations going forward out of this, neither of which are particular to the model or the analysis, but instead to the data in general.

Most of this information is overly broad in the way of being able to make determinations about patient care and treatment. Much of the data is irrelevant to treatment, at least in any but the most unusual and individualized circumstances, and the data that is there is largely about existing diagnoses as a yes/no consideration. The original rationale for this dataset, at least as it was communicated in D206, is that this is supposed to be helpful for determining why patients are readmitted. That likely requires a far, far more detailed dataset, and it would need to be much more concentrated on patient healthcare information. This comes with some concerns regarding healthcare information privacy and security, but getting into issues such as treatment types, initial complaint, number of nurse visits, etc. would likely be much more useful than many of the variables presented in this dataset.

Even if an expanded dataset is not forthcoming due to privacy concerns, I would strongly recommend that the hospital system stop tracking patient income, entirely. I've harped on this in the previous several projects because hospitals are supposed to treat people based on the fact that we as a society believe it is the right thing to do, not because of an individual's affluence or ability (or inability) to pay. At no point should there be a conversation about a patient's income in the course of determining an appropriate treatment for a problem or in assessing what care they should be given. This analysis demonstrated that it has no bearing on the length of a patient's hospitalization, which is a good thing, because it demonstrates that the patient's income isn't dictating their care. It also underlines that this data is entirely superfluous - because it has no bearing or impact, it is a waste of time to track. Furthermore, I would argue that the very fact that it is collected at all is likely to generate liability for the hospital system, because if something is collected, it can be presumed to be collected for a reason (otherwise, it would be easier to simply not collect it). This opens up the possibility for a dissatisfied patient or their family to potentially pursue a legal case against the hospital alleging that they or their loved one received suboptimal care based solely upon their income. This would be a public relations nightmare for the hospital system, and critically, fighting such a case *would require the hospital system to demonstrate that the metric has no value*, anyways. Collecting this data offers no meaningful benefits, while creating potential negative consequences, so it should be ceased immediately and retroactively removed from prior patient records.

G: Panopto Recording

My presentation of this performance assessment can be viewed here, via Panopto.

H: Code References

William Townsend D206 Performance Assessment Submission was used for the code to clean up columns in the dataset.

Mark Keith's Machine Learning in Python course materials - Intro to MLR/OLS was used to help develop the initial Multiple Regression model.

(WGU Courseware Resources was used for assistance with checking for multicolinearity by using the Variance Inflation Factor.

Mark Keith's Machine Learning in Python course materials - MLR, OLS, standardization, normalization was used for normalizing all of the data in the regression dataframe based upon the minimum/maximum for each variable.

Neal @ Tech Help Notes was used for finding the residual standard error of a regression, allowing me to compare my two multiple regression models.

GeeksForGeeks: Generating Residual Plots in Python was used for generating the residual plots of the reduced multiple regression model.

I: Source References

Statology: Multiple Linear Regression Assumptions was used to help clearly break down the assumptions inherent to a multiple regression analysis.

WGU Courseware Resources was used for some assistance with one hot encoding, as well as clarifying some of the particular requirements of this performance assessment in general.

Ashutosh Tripathi in Towards Data Science, 2019 was used to clearly describe feature selection processes for reducing the multiple regression model.

WGU Courseware Resources was used for some clarity on exactly how the coefficients are expected to be explained for the purposes of this performance assessment.