Performance Assessment 1

D208 – Predictive Modeling

Jessica Hosey

MSDA, College of Information Technology

Western Governors University

June 8th, 2024

Part I: Research Question

A.1. Relevant Research Question:

What are the main factors (symptoms, vitals, etc.) of a patient that increases the length of their hospital stay?

A. 2. Define the goals of the data analysis.

Using a multiple regression model to determine the factors (independent variables, things that are changed, explanatory, etc.) that most significantly influence the length of a patient's hospital stay (dependent variable, changes based on the independent variable, target, etc.). If these main factors are identified and are consistently seen, a hospital can use this information to provide better care to patients with these symptoms. Ultimately, if a patient is cared for well enough when presenting with these critical factors, the need for additional emergency visits (readmissions) may be reduced. This would open more time for patients who receive routine care. In addition to making patients' lives longer (and bills smaller), this will lower the hospital's chances of negligence and malpractice lawsuits. Think of it as an additional layer of protection to ensure patients receive the correct treatments at the right time.

Part II: Method Justification

B. 1. Summarize four assumptions of a multiple linear regression model.

1. a relationship between the x and y variables (dependent and independent variables) must exist. Without a relationship, there is no point in creating a linear regression model, as the variables do not correlate.

2. Multicollinearity must not exist between any of your independent variables, which means that your independent variables cannot be nearly identical. If your independent variables are similar, it will mess with the mean, standard error, etc. It will not create an environment that will help the business predict positive outcomes with your model.

3. All observations must be independent, meaning that no two data points of the independent variable can correspond. This would cause multicollinearity within your linear regression model and render it useless.

4. The residuals need to have a normal distribution or homoscedasticity. With homoscedasticity, the model will be simple, increasing the preciseness of its predictions.

B. 2. Describe two benefits of using Python to support various analysis phases.

During this analysis, using Python will be extremely helpful in creating visualizations, calculating significance, and developing our linear regression model. Python is a coding language used for various modalities in the data world. One benefit is that it supports data science processes, which is what I will be using it for in this project. In addition to its number of libraries/packages, it is effortless to learn and build upon. One specific benefit I appreciate about learning Python is that the format of how you write the code rarely changes. Some of the packages I will be using are:

- Pandas – loading and handling of the dataset

- NumPy – allows the use of mathematical calculations and reformatting of values in the dataset.
- Seaborn and Matplotlib are for the many different visualizations that can be created for this project.
- SciPy, statsmodels – for creating the multiple regression model, visualizing the residuals, evaluating the multicollinearity, and calculating the Variance Inflation Factor.
- Sklearn, preprocessing – for the transformation of the data.

B. 3. Explain why multiple linear regression is appropriate for analyzing the research question.

As stated above, the research question is "What are the main factors (symptoms, vitals, etc.) of a patient that increases the length of their hospital stay?" This question is excellent for multiple regression modeling because the relationship we analyze is between a continuous variable as its dependent variable and hospital stay length, with many continuous/categorical variables as its independent variables. Multiple linear regression demonstrates the relationship between independent and continuous dependent variables. The independent variables explain why the dependent variable (hospital stay length) is changing/increasing/decreasing.

Part III: Data Preparation

C. 1. Describe your data cleaning goals and the steps used to clean the data to achieve the goals.

The dataset is mostly clean and ready to create our model; however, some changes are needed first. Some variables are in string or Boolean form, and we need them to be numerical for our regression model. Area, Timezone, Martial, Initial_admin, Complication_risk, and Services were changed to a string using `.astype("category")`. In addition, Gender was changed to a String using the above code followed by Boolean mapping. This ensured a Yes or No response was given a numeric value. The following columns were changed from a String to a Boolean, using the code `.map(bool_mapping)`: ReAdmis, Soft_drink, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, and Asthma.

Following these changes, various validation code statements will be used to verify that the dataset is clean (no null values, duplicates, errors, etc.) and ready to create our model—examples of these code statements such as `.head()` and `.dtypes`.

C. 2. Describe the dependent variable and all independent variables using summary statistics.

Dependent variable (y) - Length of Stay (initial_days). A screenshot of the Summary Statistics is below.

Summary: Most patients spent approximately 34 days after initial admission to the hospital.

```
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
```

Independent Variables (x) - Children, Age, Income, Gender, Vitamin D levels, number of doctor visits, reason for admission, risk of complications, Arthritis, Diabetes, back pain, and amount charged. Screenshots of the summary statistics for the Independent Variables are below.

Summary: Most patients that visit the doctor had 0-1 children.

```
df.Children.value_counts().sort_index()

Children
0     2548
1     2509
2     1475
3     1489
4      995
5      169
6      191
7      213
8      209
9      108
10      94
Name: count, dtype: int64
```

```
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
```

Summary: Most patients were approximately 53 years old.

```
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
```

Summary: Most patients earn, on average, 40,000 dollars a year.

```
df.Gender.value_counts()

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

Summary: Most patients identified as female.

```
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
```

Summary: Most patients had a Vitamin D level of 18.

```
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
```

Summary: Most patients visited the doctor approximately five times, with the most visits being made nine times.

```
df.Initial_admin.value_counts().sort_index()
```

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

Summary: Most patients were admitted to an emergency visit.

```
df.Complication_risk.value_counts().sort_index()
```

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

Summary: Most patients had a medium complication risk when they were admitted.

```
df.Arthritis.value_counts()
```

```
Arthritis
No     6426
Yes    3574
Name: count, dtype: int64
```

Summary: Most patients did not report having Arthritis.

```
df.Diabetes.value_counts()

Diabetes
No     7262
Yes    2738
Name: count, dtype: int64
```

Summary: Most patients did not report having Diabetes.

```
df.BackPain.value_counts()

BackPain
No     5886
Yes    4114
Name: count, dtype: int64
```

Summary: Most patients did not have back pain.

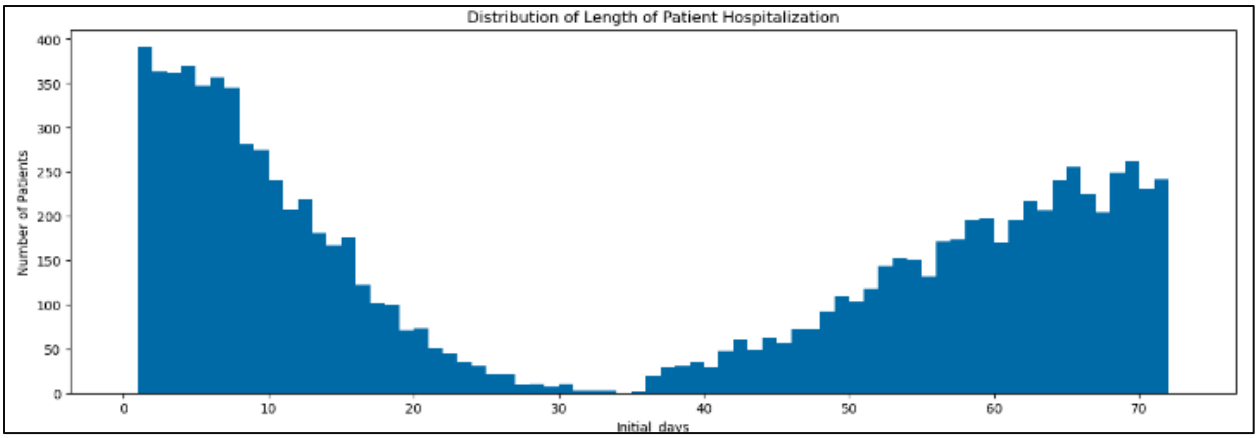
```
df.TotalCharge.describe()

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
```

Summary: Most patients were charged, on average, $5,312.17 for their doctor visits. This is incredibly high, and above, we did see that most patients were emergency admissions, so that connection helps understand the high charge.

C. 3. Generate univariate and bivariate visualizations of the distributions of the dependent and independent variables.

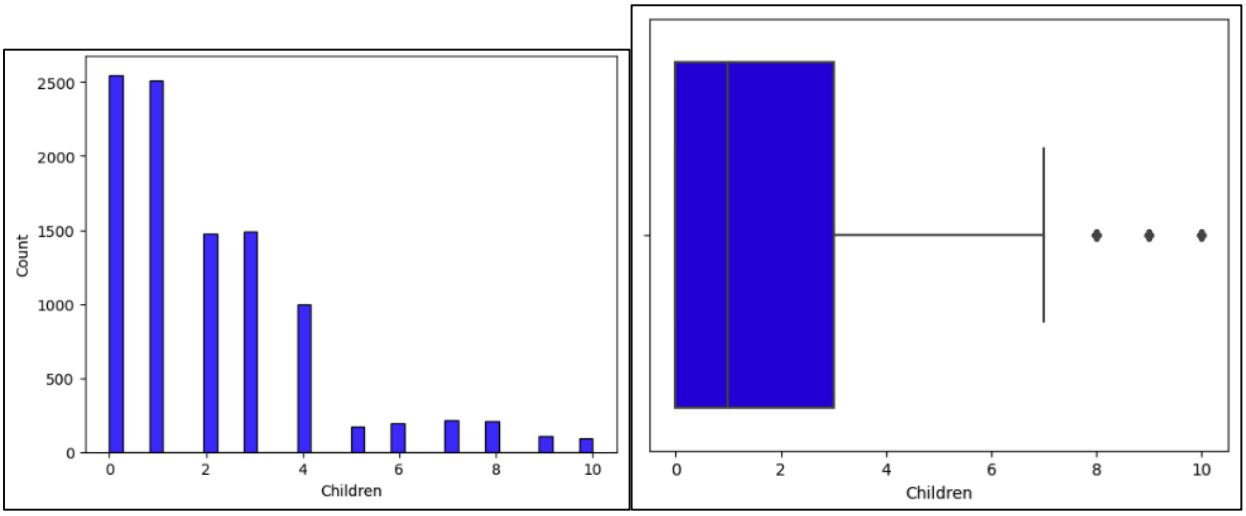
Univariate Visualizations:

Dependent Variable – Initial_days – How long have patients stayed at the hospital after initial admission?

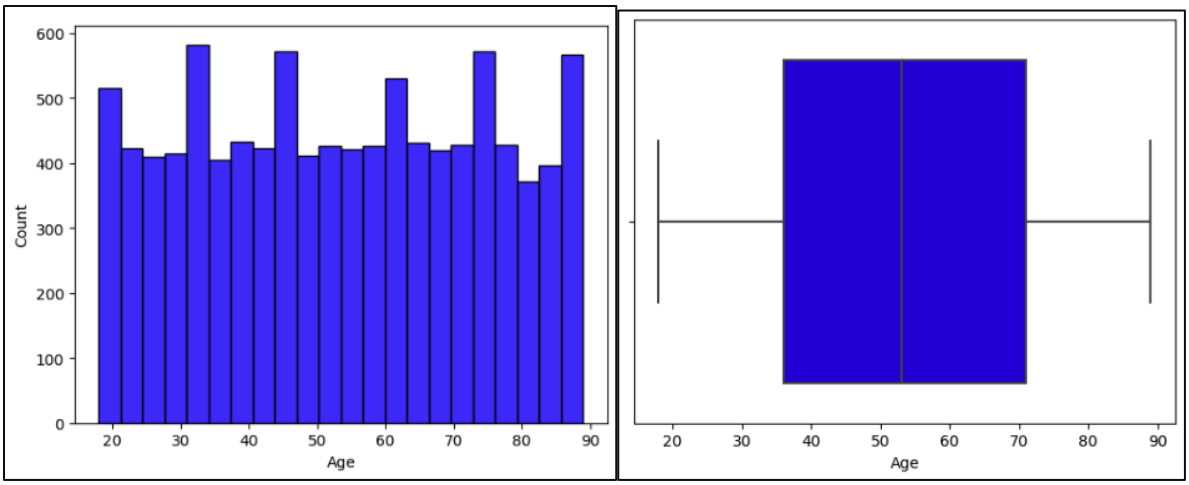


Independent Variables -

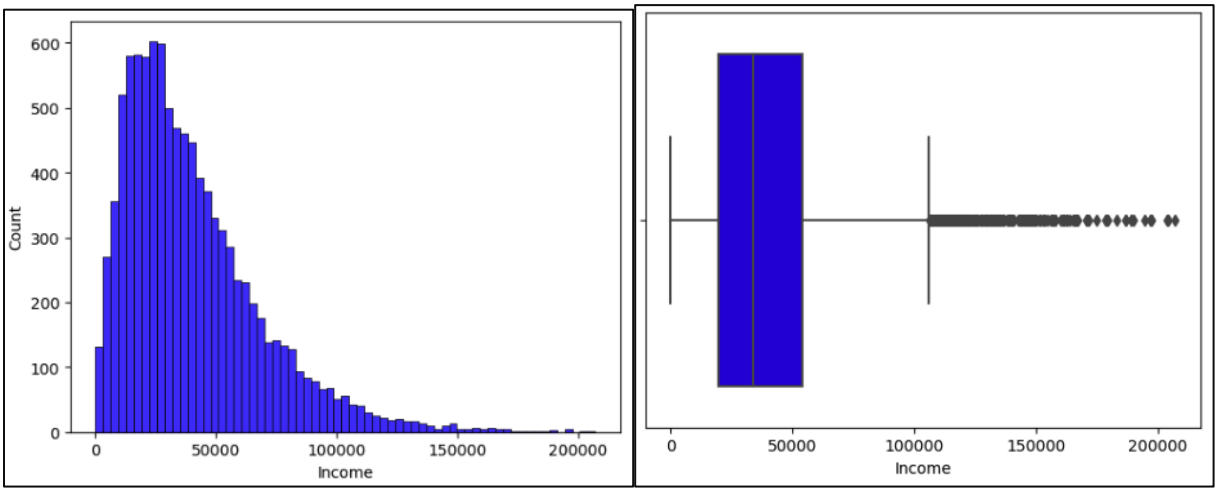
- Histograph and Boxplot – Children



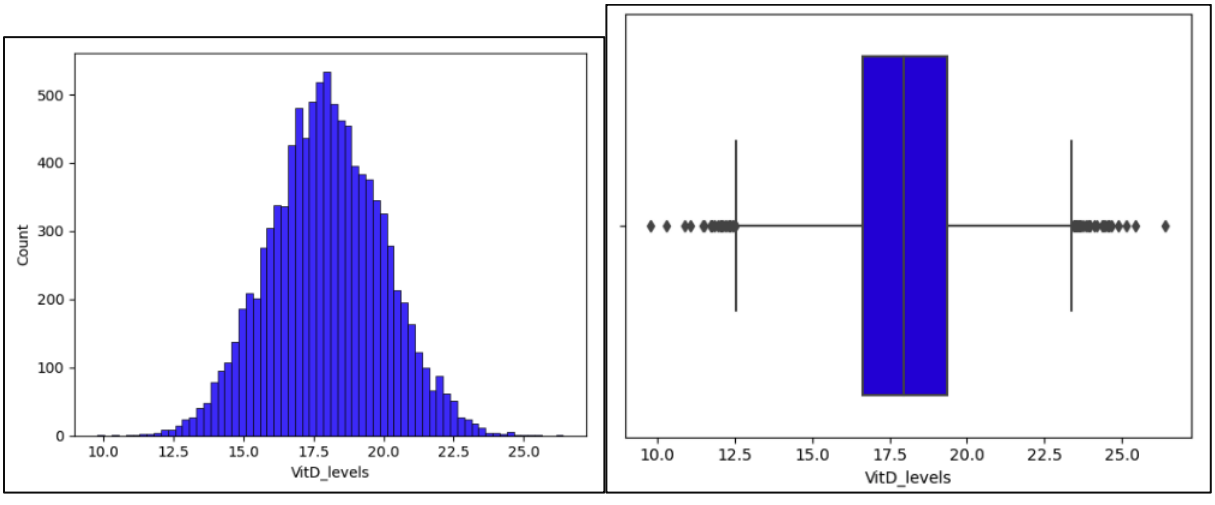
- Histograph and Boxplot – Age



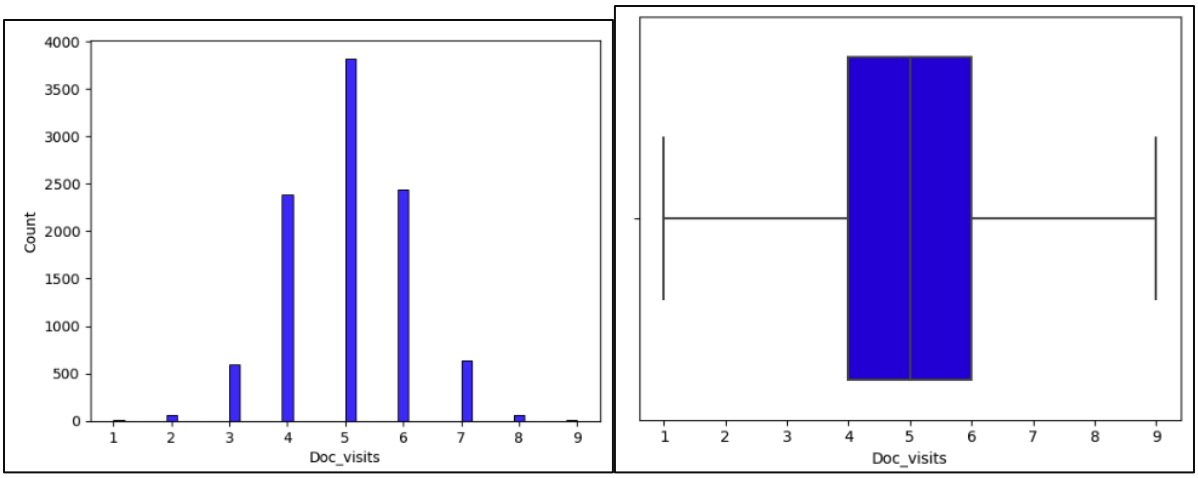
- Histograph and Boxplot – Income



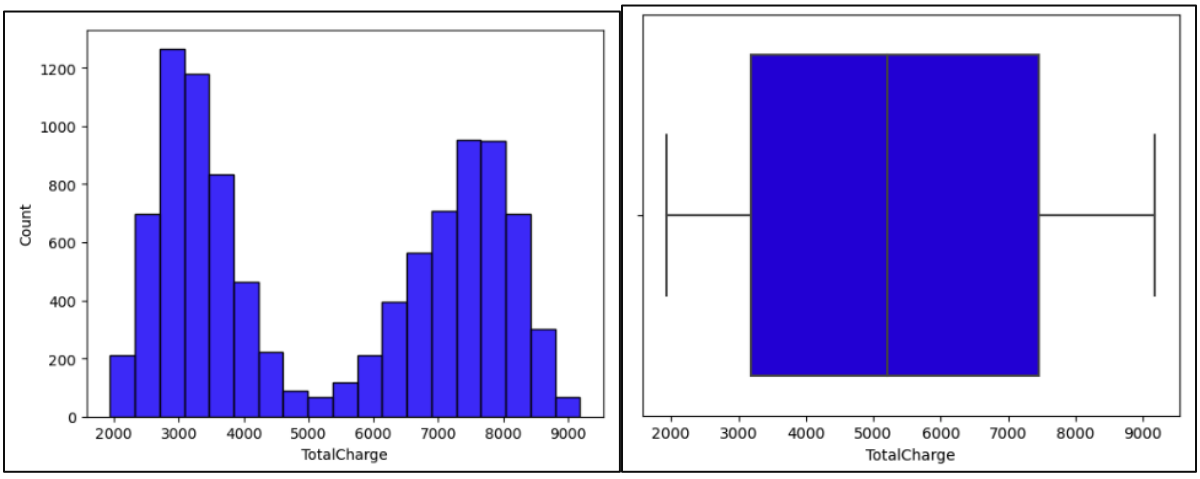
- Histograph and Boxplot – VitD_levels



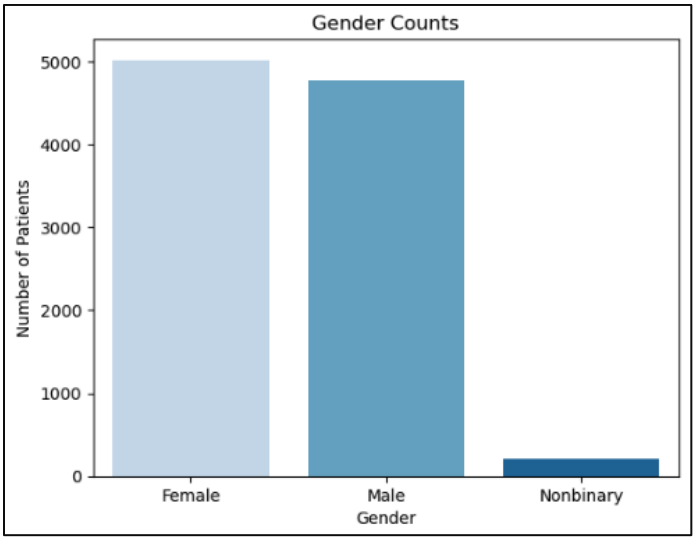
- Histograph and Boxplot – Doc_visits



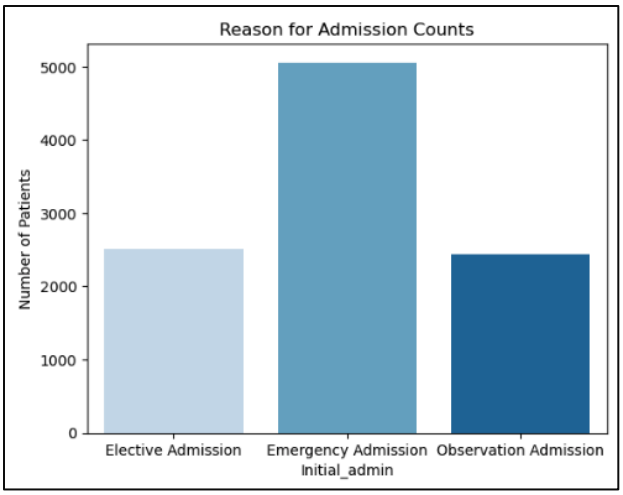
- Histograph and Boxplot – TotalCharge



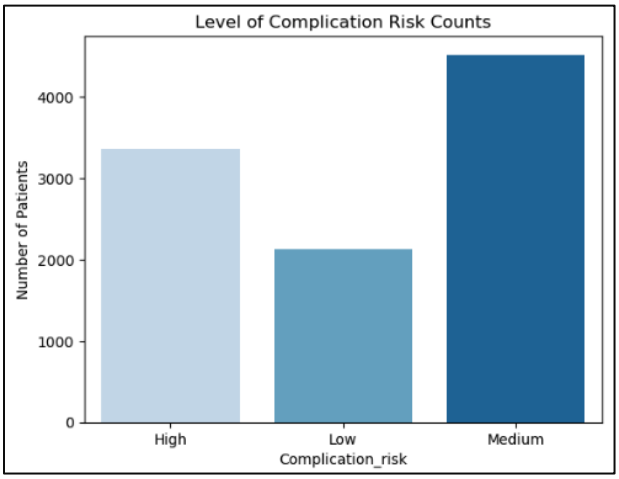
- Count Plot for Gender



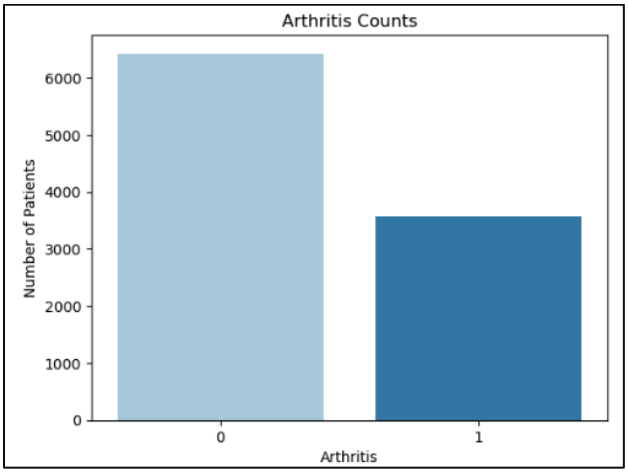
- Count Plot for Initial_admin



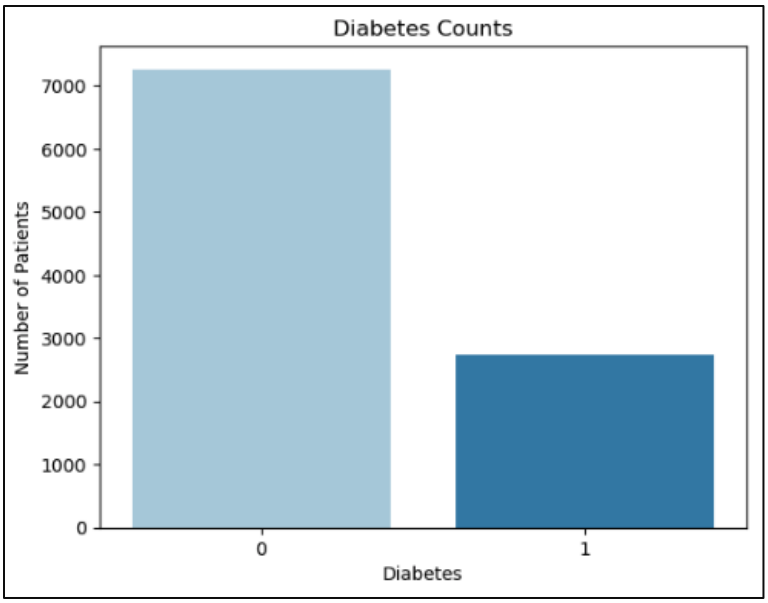
- Count Plot for Complication_risk



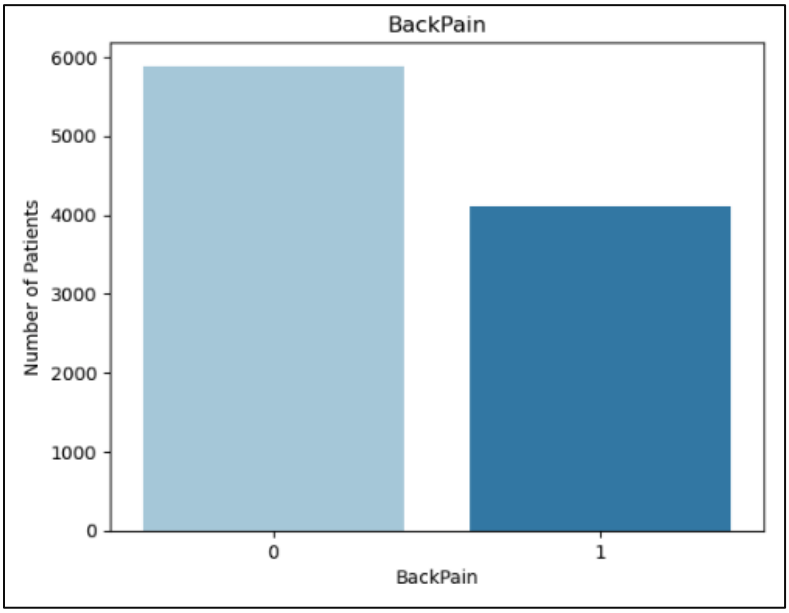
- Count Plot for Arthritis



- Count Plot for Diabetes

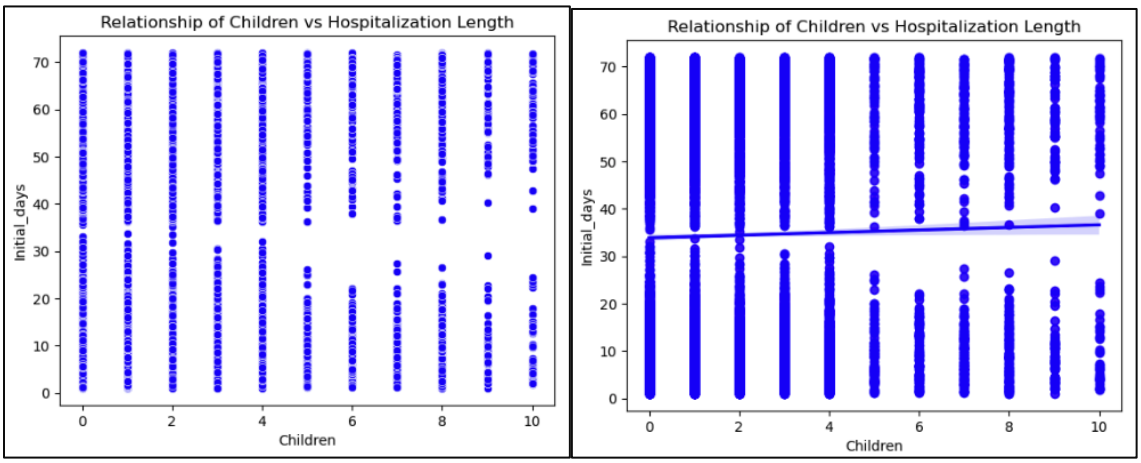


- Count Plot for BackPain

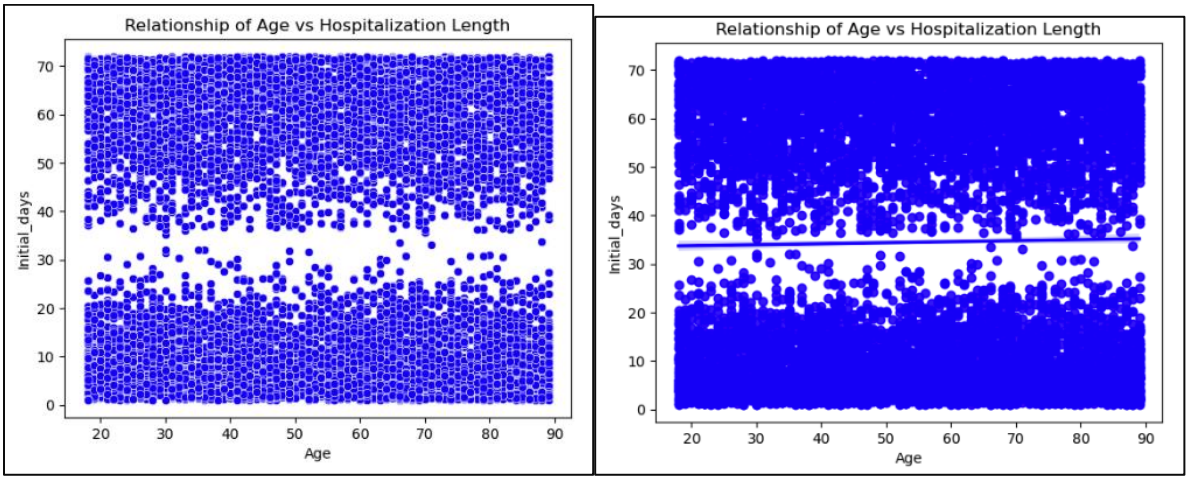


Bivariate Visualizations:

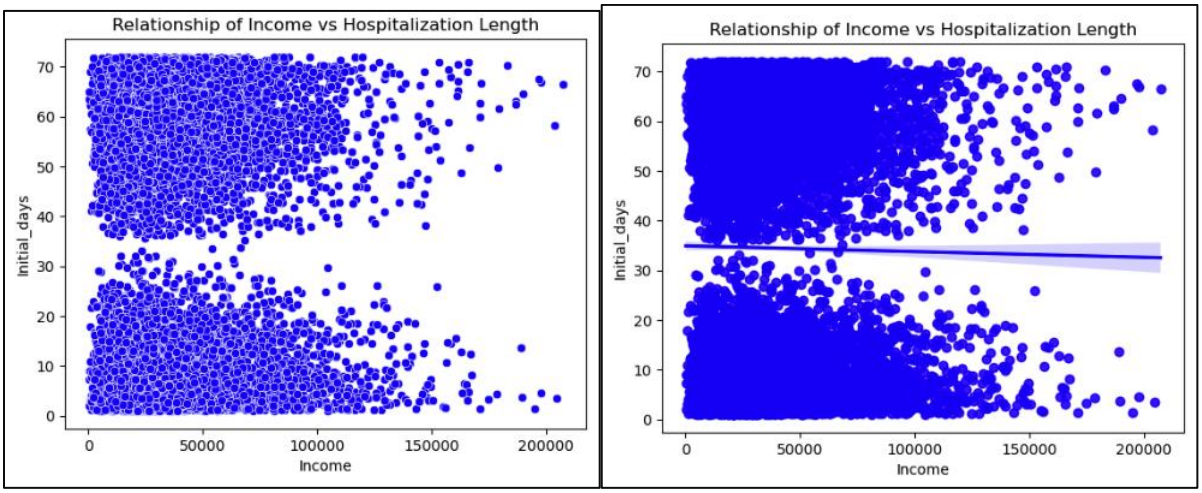
- Scatterplot and Regression Plot – Children vs Initial_days



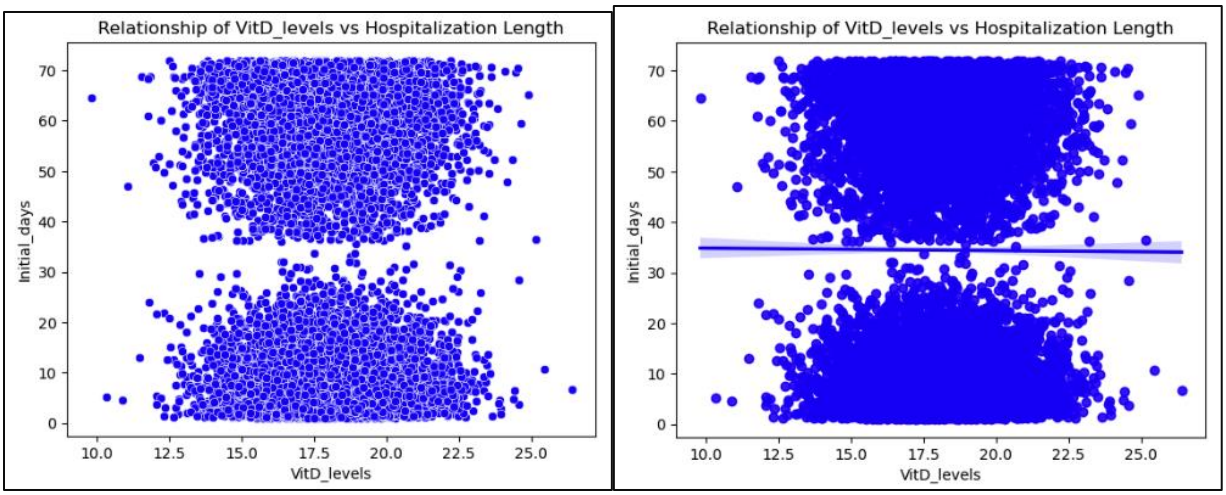
- Scatterplot and Regression Plot – Age vs Initial_days



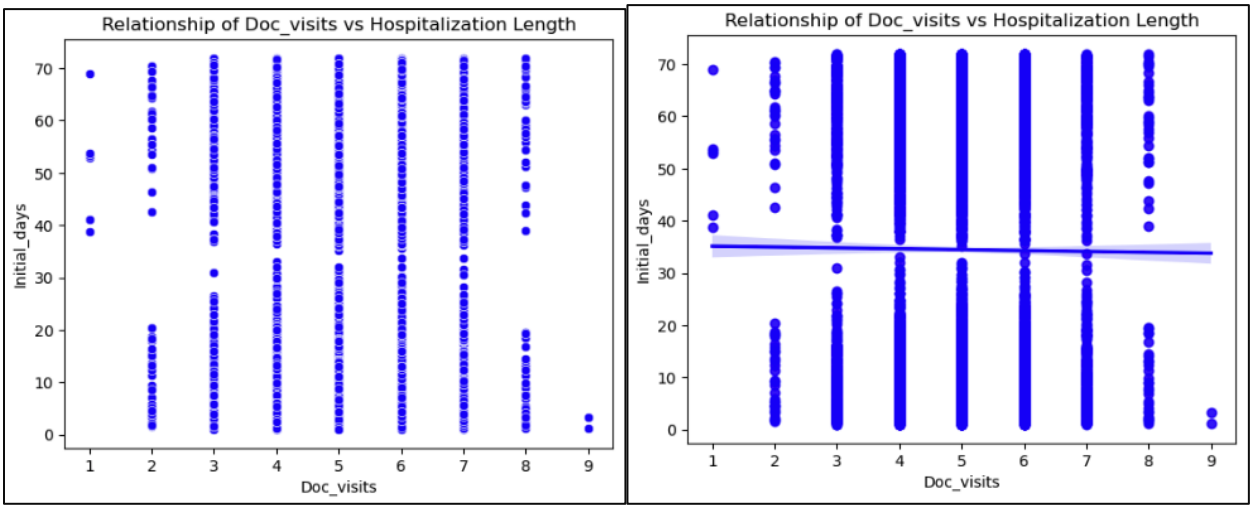
- Scatterplot and Regression Plot – Income vs Initial_days



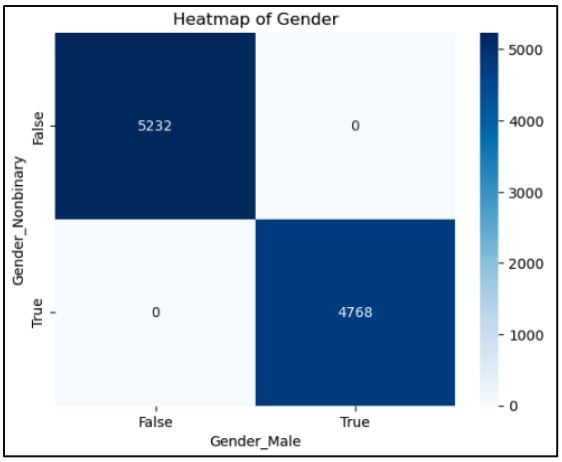
- Scatterplot and Regression Plot – VitD_levels vs Initial_days



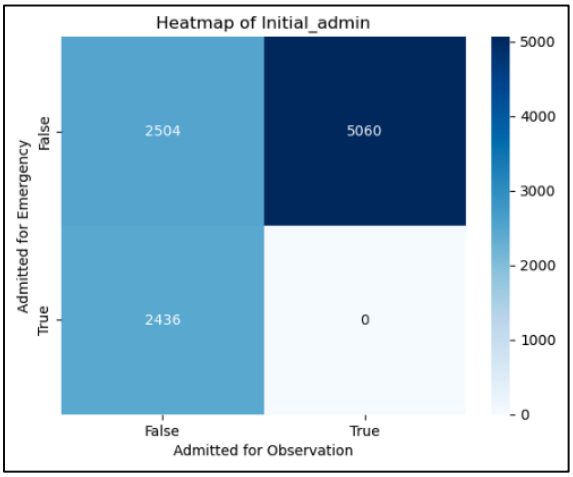
- Scatterplot and Regression Plot – Doc_visits vs Initial_days



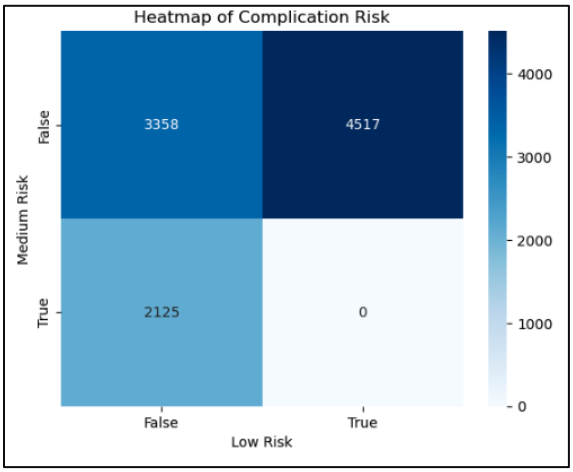
- Heatmap – Gender



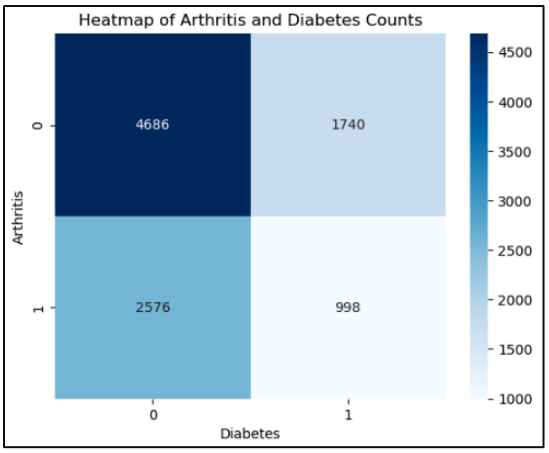
- Heatmap – Initial_admin



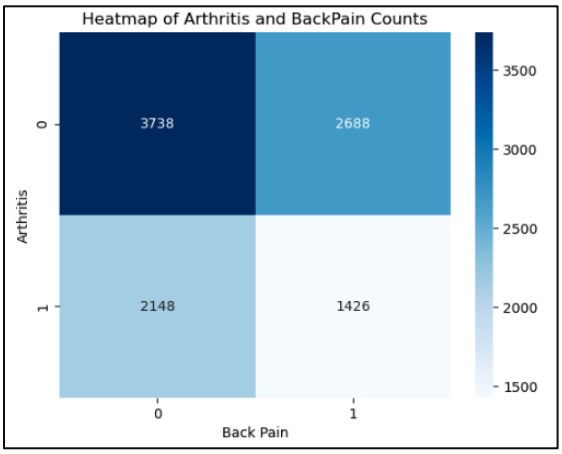
- Heatmap – Complication_risk



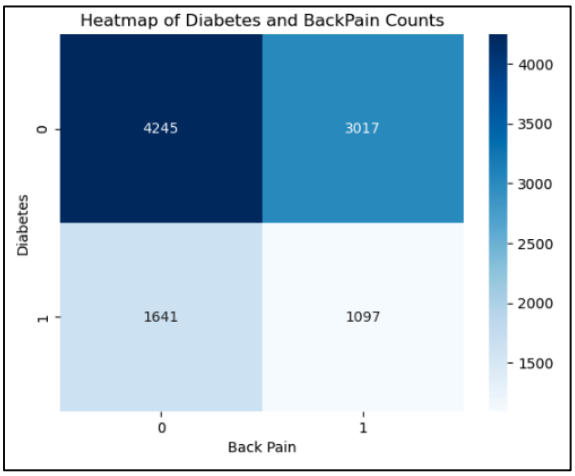
- Heatmap – Arthritis and Diabetes



- Heatmap – Arthritis and BackPain



- Heatmap – Diabetes and BackPain



C. 4. Describe your data transformation goals that align with your research question and the steps used to transform the data to achieve the goals.

The Boolean variables will be changed from true/false to 1/0, and the string values will be handled differently based on whether they are ordinal or nominal:

The ordinal categorical variables will be modified to give more information. Currently, there is no relationship between "1" and "8" in the dataset. However, humans understand that the relationship between those values is that "1 is considered 'most important' and eight is considered 'least important'" to the patient from a survey. We need to provide the data frame a statement that gives more value to those numbers in those columns, such as "1 < 8". This way, the value has more meaning in the dataset, and if we tried to use this information to calculate significance, the computer would know that "8" has less value or weight than a "1" would in these columns. I used the following code snippets to establish this numerical relationship for the Item 1 through Item 8 survey responses: `CategoricalDtype`, `.map`, and `.astype`.

As for nominal categorical variables, these will generate "dummy columns" to represent the data as binary numbers. This means a two or three options variable such as whether the patient has Diabetes. The values in this column are either "Yes or No" and can be remapped into a numerical form of "1 or 0". The following code established the numeric relationship for Yes and No responses: `pd.get_dummies`, and `.insert`.

After initial cleaning, there was the need to use another step to ensure that all the data in the independent variables was numeric before I started to create my model. I still had some non-numeric values and needed to change them. The following code was used to convert all data to numeric: `convert_to_numeric`, `.columns`, `.dtype`, `.loc`, `.astype`, and `.to_numeric`. These code snippets created a function that checked the dataset for Boolean values that needed to be converted to integers.

C. 5. Provide the prepared data set as a CSV file.

See the uploaded files that were submitted for Evaluation.

Part IV: Model Comparison and Analysis

D.1. Construct an initial multiple linear regression model.

```
#Initial Model
#Set dependent variable for Y
y = regression_df.Initial_days

#Set multiple independent variables for X and add constant
X = regression_df[expected_columns]
X = sm.add_constant(X)  # This will add a constant column

#Double checking that all X columns are numeric
X = X.apply(pd.to_numeric, errors='coerce')

#Fit and print the model
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                          OLS   Adj. R-squared:                  0.998
Method:               Least Squares   F-statistic:                 3.986e+05
Date:              Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                      17:21:40   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]
------------------------------------------------------------------------------
const            -29.4951      0.124   -237.141      0.000     -29.739     -29.251
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
VitD_levels         0.0029      0.006      0.525      0.600      -0.008       0.014
Doc_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
BackPain           -1.0633      0.023    -46.947      0.000      -1.108      -1.019
TotalCharge         0.0122   5.16e-06   2359.774      0.000       0.012       0.012
```

```
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.                     6.21e+20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.4e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

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

```
1.1126862370663226
```

D. 2. Justify a statistically based feature selection procedure or a model evaluation metric to reduce the initial model.

Due to the warning at the bottom of the model's output, identifying and eliminating variables is recommended, as there are issues with the model due to multicollinearity. As stated above in B1, one of the assumptions of multiple regression models is that our variables must not be too correlated to another variable (Reed). This issue will cause problems with the standard error and render the variable statistically insignificant if it is not removed from the model. If two variables are highly correlated, they are nearly identical, inflating the mean and stopping us from seeing the difference between them. Hence, this stops us from seeing what variables influence our research question.

The two ways that I chose to select variables to eliminate are the Variance Inflation Factor and P-values. The Variance Inflation Factor (VIF) is used to analyze how much an independent variable influences the standard error value of the model. A variable with a high VIF value must be removed from the model to ensure the standard error is not inflated. I removed the first two variables due to the high VIF value below. Both VitD_levels and Doc_visits were well over a VIF of 10.

```
#Warning for multicollinearity - check for VIF to see if variables need to be eliminated
X = regression_df[["Children", "Age", "Income", "gender_male", "gender_nonbinary", "VitD_levels", "Doc_visits",
                   "initial_admit_observ", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "Arthritis",
                   "Diabetes", "BackPain", "TotalCharge"]]

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                Children   1.932741
1                     Age   7.361227
2                  Income   2.963630
3             gender_male        inf
4        gender_nonbinary        inf
5              VitD_levels  29.950450
6               Doc_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               BackPain   1.697394
14           TotalCharge   6.760867
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/statsmodels/stats/outliers_influence.py:198: RuntimeWarning: divide by
zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

After removing those two variables, as stated above, I reran the model, and a warning for multicollinearity still was given. So, in addition to VIF values, I used P-values to eliminate any independent variables that may influence the model's standard error further. Any independent variable with a p-value over 0.05 was eliminated. Looking at the second screenshot below, there is a column for P-values (P>[t]). We can see that the columns Children, Age, Income, and Gender have the highest p-values. I eliminated those values individually, and the warning for multicollinearity disappeared. Also, using P-values to eliminate variables helps increase the preciseness of our numbers and the homoscedasticity of the data. The more homoscedasticity the model has, the more simplified it is, and the more well it fits to produce reliable predictions from the data. Variables above 0.05 P-value contribute to the significance of the independent variable to the dependent variable. This would render my model's predictions invalid.

```
#Backwards Elimination 1: Seek highest p-value above 0.05
y = regress_df_minmax.Initial_days
X = regress_df_minmax[["Children", "Age", "Income", "gender_male", "gender_nonbinary",  "initial_admit_observ", "initial_admit_emerg",
                       "comp_risk_low", "comp_risk_medium", "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 4.651e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        17:23:25   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]
------------------------------------------------------------------------------
const         -0.0957      0.001   -149.616      0.000      -0.097      -0.094
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
BackPain              -0.0150      0.000    -46.950      0.000      -0.016      -0.014
TotalCharge            1.2428      0.001   2359.944      0.000       1.242       1.244
==============================================================================
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.                     2.31e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.83e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

D.3. Provide a reduced linear regression model that follows the feature selection or model evaluation process in part D2.

Eliminating VitD_levels and Doc_visits as their VIF values are the highest and above 10.

```
#Eliminated VitD_levels (VIF = 29.950450) and Doc_visits (VIF = 19.835725), running VIF again to see if any values are still above 10
X = regression_df[["Children", "Age", "Income", "gender_male", "gender_nonbinary",  "initial_admit_observ", "initial_admit_emerg",
                   "comp_risk_low", "comp_risk_medium", "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]

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              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             BackPain  1.670107
12          TotalCharge  5.435098
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/statsmodels/stats/outliers_influence.py:198: RuntimeWarning: divide by zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

Using P-Values to Reduce the Model Further:

```python
#Backwards Elimination 1: Seek highest p-value above 0.05
y = regress_df_minmax.Initial_days
X = regress_df_minmax[["Children", "Age", "Income", "gender_male", "gender_nonbinary",  "initial_admit_observ", "initial_admit_emerg",
                  "comp_risk_low", "comp_risk_medium", "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

OLS Regression Results

Dep. Variable:	Initial_days	R-squared:	0.998
Model:	OLS	Adj. R-squared:	0.998
Method:	Least Squares	F-statistic:	4.651e+05
Date:	Wed, 05 Jun 2024	Prob (F-statistic):	0.00
Time:	17:23:25	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]
const	-0.0957	0.001	-149.616	0.000	-0.097	-0.094
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
BackPain	-0.0150	0.000	-46.950	0.000	-0.016	-0.014
TotalCharge	1.2428	0.001	2359.944	0.000	1.242	1.244

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.	2.31e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.83e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Elimination of both Gender Variables as the P-Values are above 0.05

```python
#BACKWARD ELIMINATION 2: Seek highest p-value above 0.05 (eliminated gender_male and gender_nonbinary, p-values of 0.946)
y = reg_df_minmax.Initial_days
X = reg_df_minmax[["Children", "Age", "Income",  "initial_admit_observ", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium",
                   "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 5.074e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        16:58:22   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]
const	-0.0957	0.001	-154.535	0.000	-0.097	-0.095
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
BackPain	-0.0150	0.000	-46.957	0.000	-0.016	-0.014
TotalCharge	1.2428	0.001	2360.121	0.000	1.242	1.244

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.

Elimination of the Children Variable as the P-Value is above 0.05

```
#Backwards Elimination  3: Seek highest p-value above 0.05 (eliminated Children, p-value of 0.816)
y = regress_df_minmax.Initial_days
X = regress_df_minmax[["Age", "Income",  "initial_admit_observ", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium",
                  "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 5.582e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        17:23:35   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]
const	−0.0957	0.001	−158.610	0.000	−0.097	−0.095
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
BackPain	−0.0150	0.000	−46.963	0.000	−0.016	−0.014
TotalCharge	1.2428	0.001	2360.835	0.000	1.242	1.244

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.

Elimination of the Age Variable as the P-value is above 0.05

```python
#Backwards Elimination  4: Seek highest p-value above 0.05 (eliminated Age, p-value of 0.585)
y = regress_df_minmax.Initial_days
X = regress_df_minmax[["Income",  "initial_admit_observ", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium",
                       "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 6.203e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        17:37:03   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]
const	−0.0958	0.001	−177.580	0.000	−0.097	−0.095
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
BackPain	−0.0150	0.000	−46.986	0.000	−0.016	−0.014
TotalCharge	1.2428	0.001	2361.202	0.000	1.242	1.244

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.

Elimination of the initial_admit_observ Variable as the P-Value is above 0.05

```python
#Backwards Elimination  5: Seek highest p-value above 0.05 (eliminated initial_admit_observ, p-value of 0.505)
y = reg_df_minmax.Initial_days
X = reg_df_minmax[["Income", "initial_admit_emerg", "comp_risk_low", "comp_risk_medium",
                   "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 6.979e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        17:37:09   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]
const	-0.0957	0.000	-192.771	0.000	-0.097	-0.095
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
BackPain	-0.0150	0.000	-46.983	0.000	-0.016	-0.014
TotalCharge	1.2428	0.001	2361.289	0.000	1.242	1.244

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.

Elimination of the Income Variable as the P-value is above 0.05.

This is the final reduced model below:

```
#Backwards Elimination  6: Seek highest p-value above 0.05 (eliminated Income, p-value of 0.367)
y = reg_df_minmax.Initial_days
X = reg_df_minmax[["initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "Arthritis", "Diabetes", "BackPain", "TotalCharge"]]
X = sm.add_constant(X)  # This will add a constant column

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                          OLS   Adj. R-squared:                  0.998
Method:               Least Squares   F-statistic:                 7.976e+05
Date:              Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                      17:37:18   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]
const	−0.0959	0.000	−217.782	0.000	−0.097	−0.095
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
BackPain	−0.0150	0.000	−46.995	0.000	−0.016	−0.014
TotalCharge	1.2428	0.001	2361.509	0.000	1.242	1.244

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.

As seen above in the P>[t] column, no independent variables with a P-value greater than 0.05 exist now.

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

```
0.015672009109192912
```

E. 1. Explain your data analysis process by comparing the initial multiple linear regression and the reduced linear regression models.

Below are the screenshots of the before and after model reduction to visually compare the two model's standard error, p-values, and residual standard error. Multiple Linear Regression is used to see how significantly the independent variables influence the dependent variable. In this research question, seeing what patients are dealing with increases the likelihood of more extended hospital stays. Looking at the initial vs. final model, we can see that many independent variables were removed due to multicollinearity and heteroscedasticity within the model. The VIF value helped to remove the multicollinear concerns, and the P-value to make the model more homoscedastic. Due to the similarity of both the initial and reduced model's P-values, high r-squared values indicate a need for additional metric to compare the models. The

use of residual standard error was completed to see the difference between the two models. When there is a lower residual standard error, there is a higher probability that the model's predictions will be more accurate and precise (Reed). The screenshots below show that the initial model's residual standard error was 1.112686, and the reduced residual standard error was 0.01567. The initial model's residual standard error was significantly reduced with the backward elimination of some independent variables. This indicates that the final reduced model is better for predicting how many days a patient might stay at the hospital than the initial model's predictions.

Initial Model:

```
#Initial Model
#Set dependent variable for Y
y = regression_df.Initial_days

#Set multiple independent variables for X and add constant
X = regression_df[expected_columns]
X = sm.add_constant(X)  # This will add a constant column

#Double checking that all X columns are numeric
X = X.apply(pd.to_numeric, errors='coerce')

#Fit and print the model
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 3.986e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        17:21:40   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]
----------------------------------------------------------------------------------------------
const                       -29.4951      0.124   -237.141      0.000     -29.739     -29.251
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
VitD_levels                   0.0029      0.006      0.525      0.600      -0.008       0.014
Doc_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
BackPain                     -1.0633      0.023    -46.947      0.000      -1.108      -1.019
TotalCharge                   0.0122   5.16e-06   2359.774      0.000       0.012       0.012
```

```
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.                     6.21e+20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.4e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

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

```
1.1126862370663226
```

Final Model:

```
#Final Reduced Model
y = regress_df_minmax.Initial_days
X = regress_df_minmax[["initial_admit_emerg", "comp_risk_low", "comp_risk_medium", "Arthritis",
                       "Diabetes", "BackPain", "TotalCharge"]].assign(const=1)

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 7.976e+05
Date:                Wed, 05 Jun 2024   Prob (F-statistic):               0.00
Time:                        17:51:27   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
BackPain               -0.0150      0.000    -46.995      0.000      -0.016      -0.014
TotalCharge             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.
```

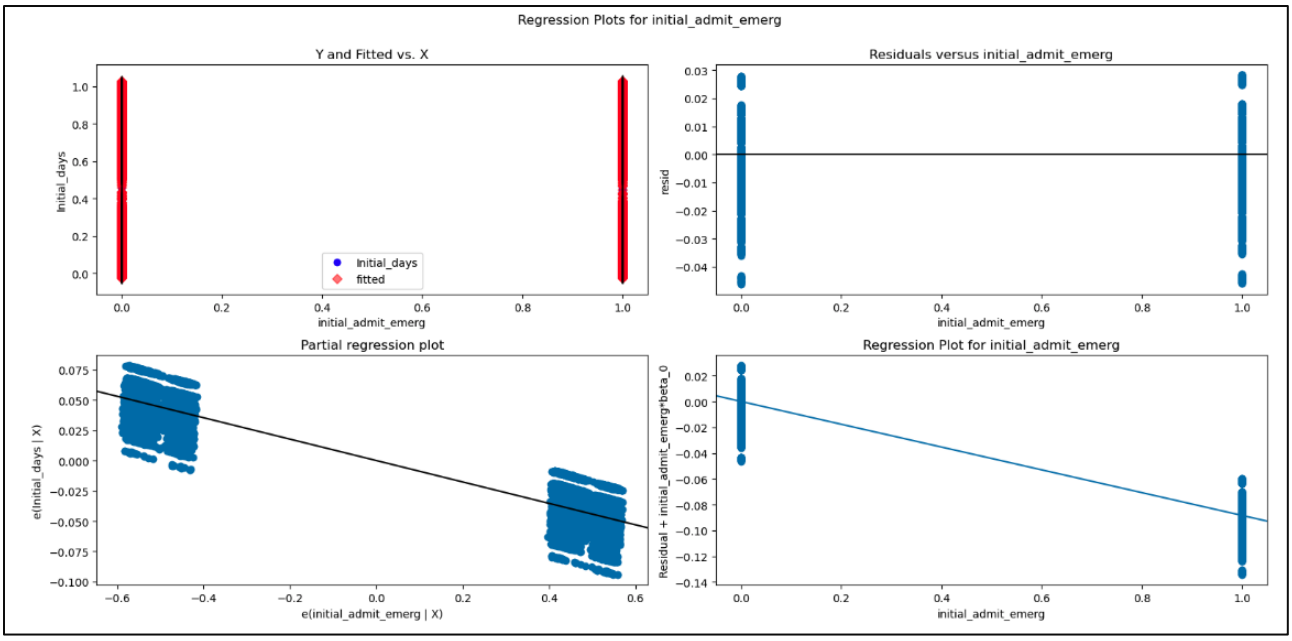
```
results.resid.std(ddof=X.shape[1])
```

```
0.015672009109192912
```

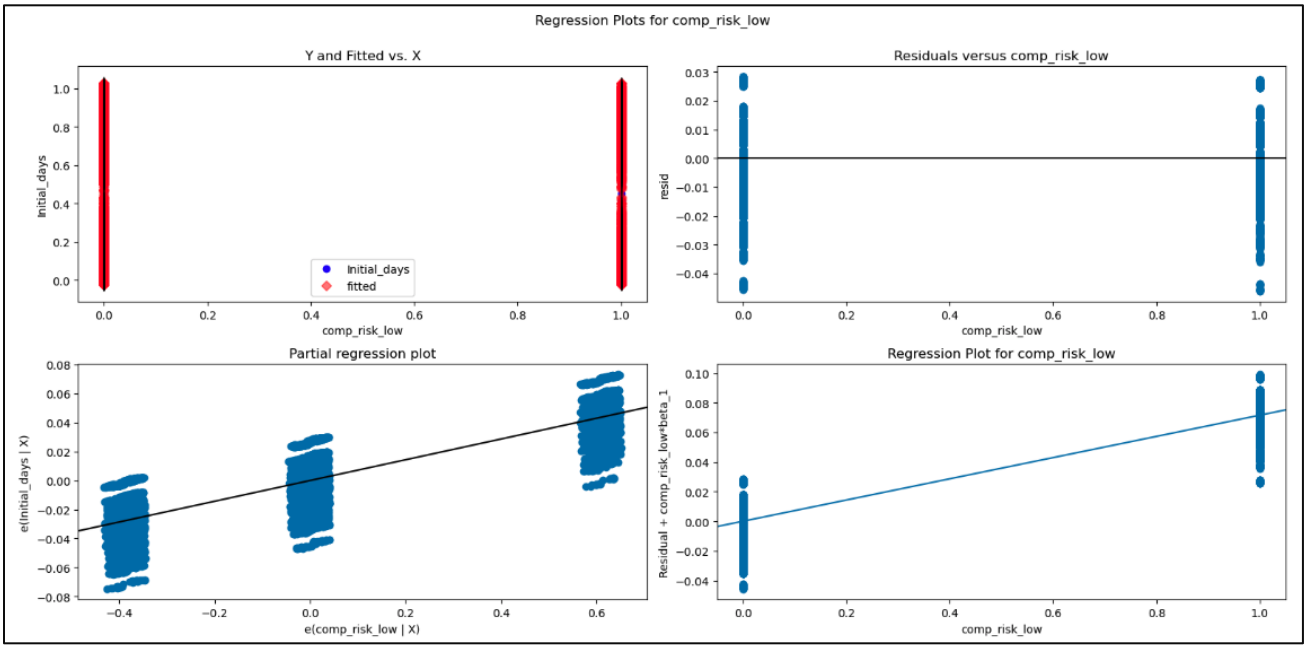
E. 2. Provide the output and all calculations of the analysis you performed.

Residual Plots

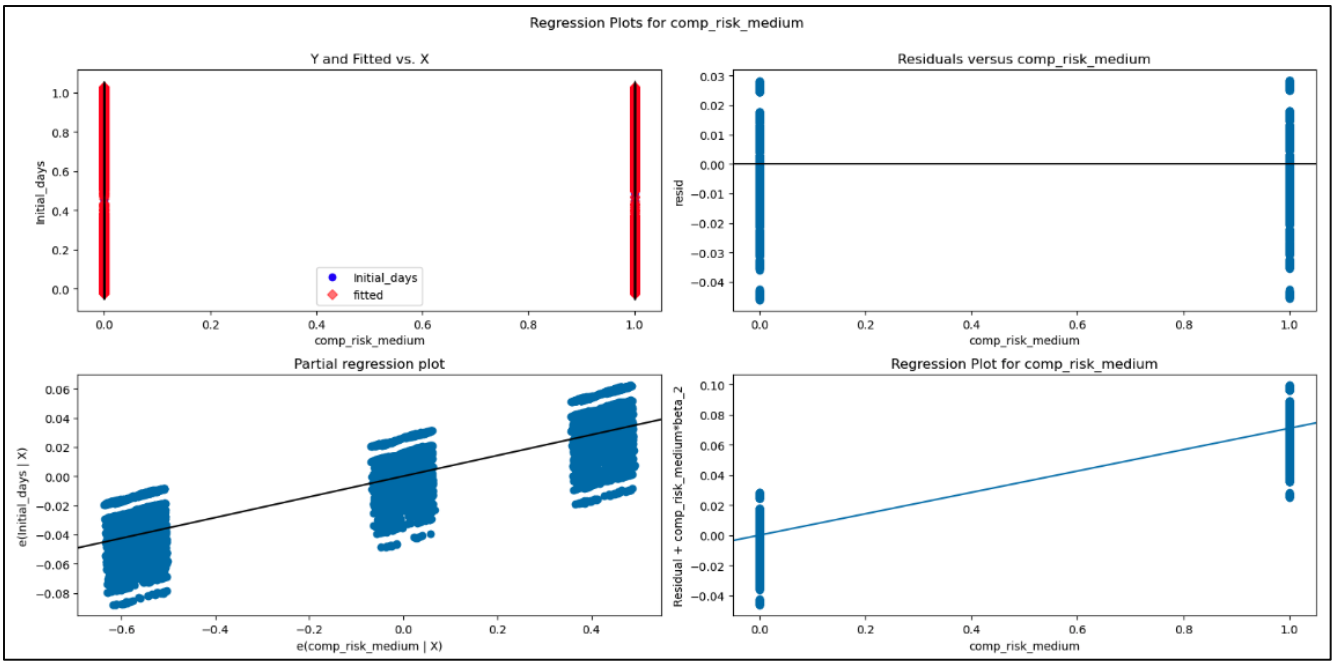
Initial_admit_emerg Regression Plot



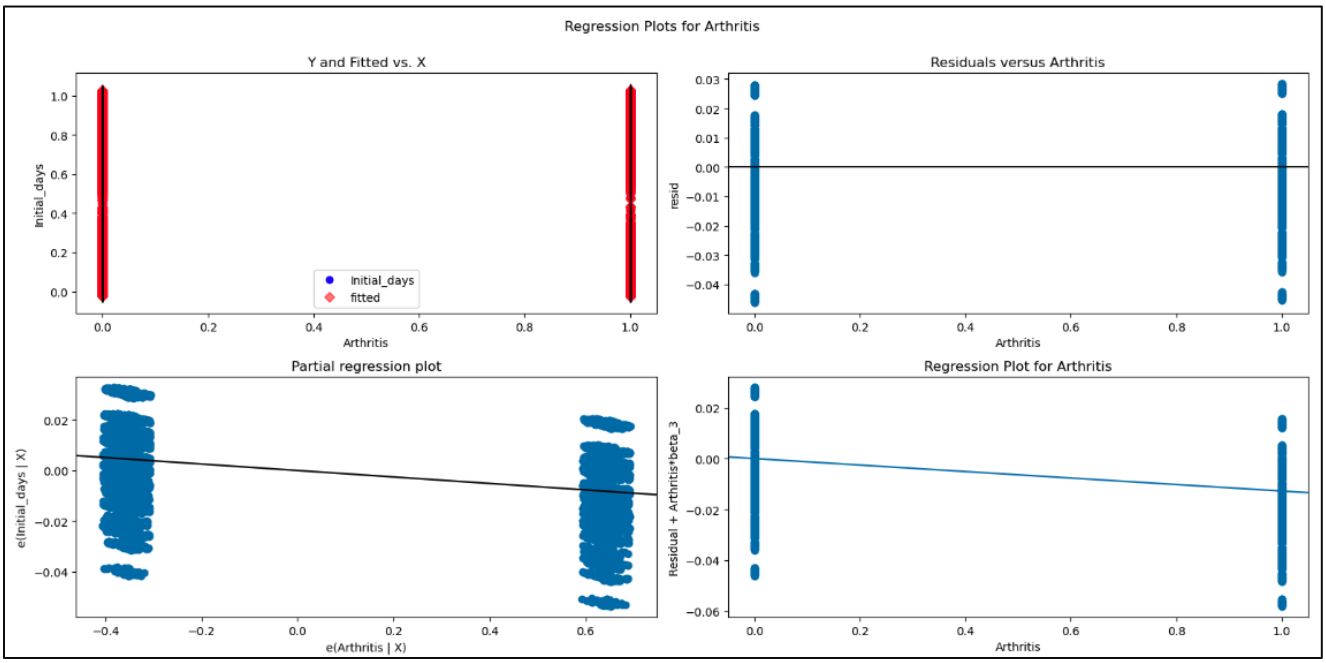
Comp_risk_low Regression Plot



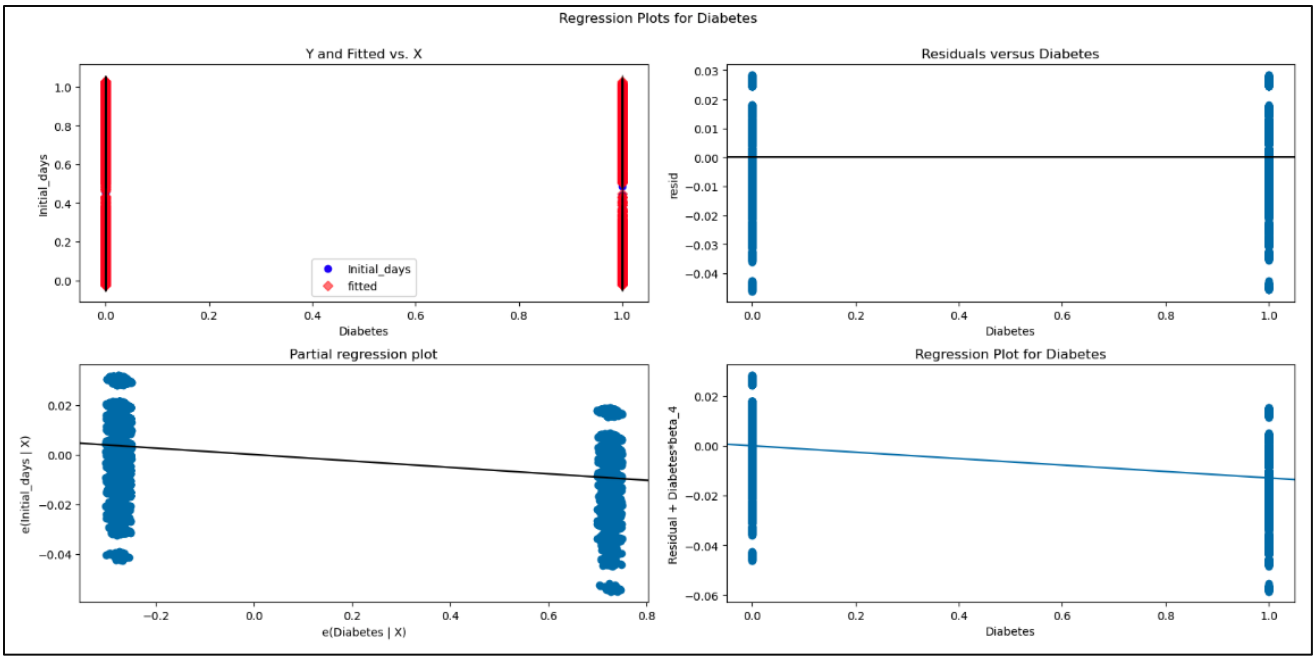
Comp_risk_medium Regression Plot



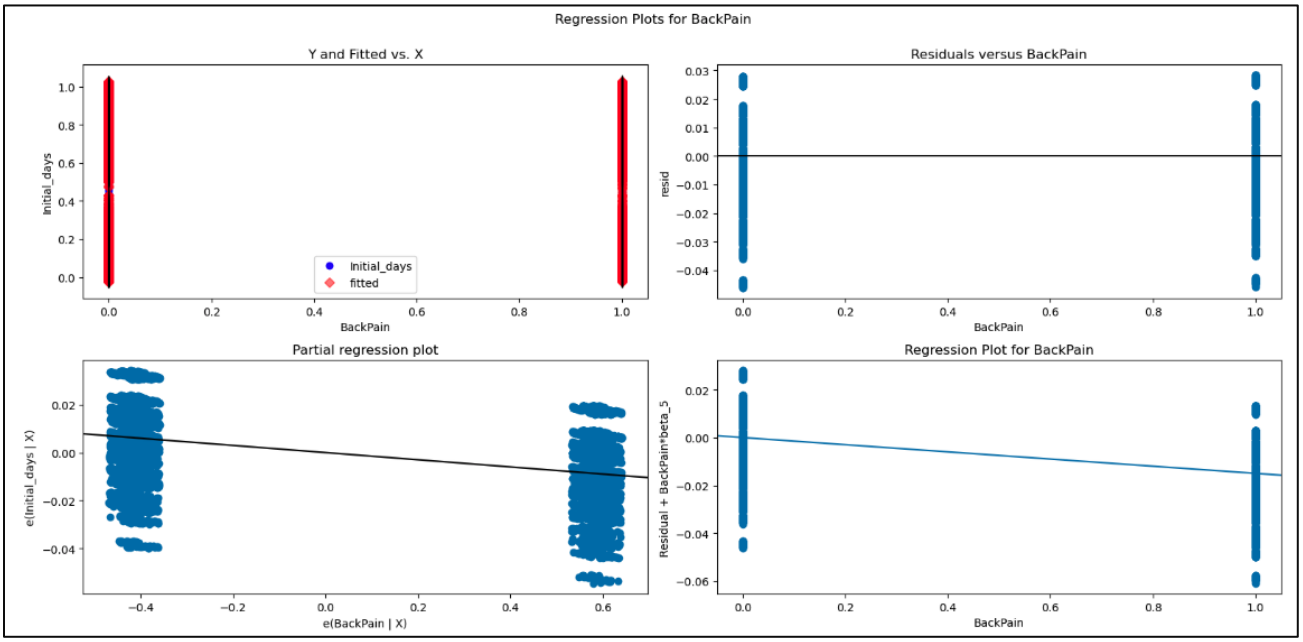
Arthritis Regression Plot



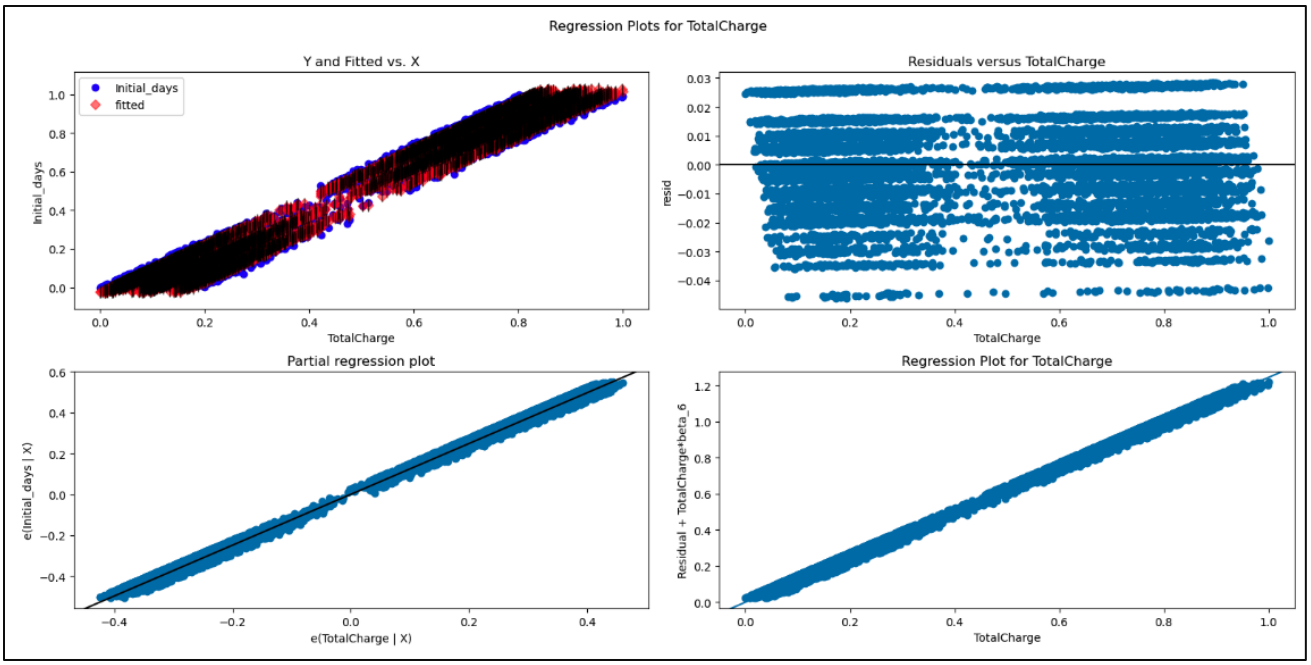
Diabetes Regression Plot



BackPain Regression Plot



TotalCharge Regression Plot



Model's Residual Standard Error

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

```
0.015672009109192912
```

E. 3. Code File:

Jupyter Notebook Code File was uploaded with the submission for Evaluation

Part V: Data Summary and Implications

F. 1. Discuss the results of your data analysis, including the following elements:

- **a regression equation for the reduced model**

y = -0.0959 (Constant) - 0.0884 *(Initial Admission - Emergency)* + 0.0716 *(Complication Risk – Low)* + 0.0709 *(Complication Risk – Medium)* - 0.0128 *(Arthritis)* - 0.0129 *(Diabetes)* - 0.0150 *(BackPain)* + 1.2428 *(TotalCharge)*

- **an interpretation of the coefficients of the reduced model**

When keeping things constant, the following observations can be seen due to the coefficients given above in the regression equation (as well as seen above in the Final Model Results):

- Patients admitted under emergency conditions spend 8.84% fewer days in the hospital.

- Patients with a low risk for complications upon admission spend 7.16% more days admitted in the hospital.
- Patients who have a medium risk for complications upon admission spend 7.09% more days admitted in the hospital.
- Any patient admitted with Arthritis spends 1.28% fewer days admitted in the hospital
- Any patient admitted with Diabetes spends 1.29% fewer days in the hospital.
- Upon admission, any patient with back pain spends 1.50% fewer days admitted in the hospital.
- If there is a Total Charge increase of 1 unit, the patient's stay at the hospital spends 1.2 additional days admitted in the hospital.

- **the statistical and practical significance of the reduced model**

The reduced model is precise and statistically significant, as all the independent P-values are below 0.05. In addition to the P-value, the f-statistic shows that the model is significant. If we had a f-statistic value above 0.05, the null hypothesis would be accepted, and there would be no significance between the independent and dependent variables. Since the model has a 0.00 F-statistic value, at least one independent variable is related to the dependent variable (Mahmood 2021). Additionally, the model's Durbin-Watson statistic value is 1.974, which is within the ideal range for this test, which means the model has low or no bias between the variables (Mahmood 2021).

As for practical significance, the model is not very useful as it looks at only a tiny amount of vitals and information that a patient may enter a hospital with at any point. Additionally, patients are human, and human actions are not always predictable. This means that the data entering a hospital varies widely on what enters their doorways. Also, patients have a choice of hospitals (most of the time).

- **the limitations of the data analysis**

- Data is from a single day of operation (24 hours). This is varied as more people are moving around on weekends than weekdays, which could cause a patient's need to be seen. Also, if the patient is employed, they do not have time to enter a hospital during the week and may only be able to visit on the weekend (or after clinic hours) in an emergency room.
- The dataset does not contain any information past intake information. Some patients' symptoms changed after that initial intake reading. Some individuals get better, and some die. Information about the individual's entire visit might help the model's ability to predict hospital stay length. Information should also be about their exit vitals and how the patient felt before leaving the hospital. This information might help to see why they are frequent hospital visitors or not.
- The dataset does include income information but does not factor in whether a patient has health insurance. This would provide a big difference in the data and may provide more insight into whether a patient is coming to the hospital as a last resort option (because they are uninsured and need help) or just because they can (insured and go to the hospital whenever).
- The dataset only contains information about one hospital. If we want to create a model to predict the cause of length of stay, it might be good to add information from all hospitals, not just a single hospital for a business.
- More information from the original dataset could have been used to evaluate hospital stays within the model. So again, this model is of a small subset of information about a patient's stay.

F. 2. Recommend a course of action based on your results.

The model is a good practice and could be used to predict a simple set of potential patient outcomes; however, it is not a good use to make business decisions as more data could be used but not available. Also, the unknown about what patients come to the hospital and at what point in their lifetimes is entirely unpredictable. An insured individual may visit the hospital more frequently for less emergency reasons versus someone who does not have insurance and visits once every ten years due to an emergency.

My recommended course of action for the hospital is to continue collecting data, gather more information about those entering their buildings, and for longer than a day (5 years would be better) to complete their dataset. With more data, the model's insights could help determine the next steps for the business. In addition to more data, the hospital needs to add more information about the patient's hospital stay, vitals before leaving, whether they have health insurance is more insightful than income, whether their visit was due to a traumatic event (car accident, etc.), information about their most recent visit,

Part VI: Demonstration

G. Panopto Video:

Panopto Video Link

H. Code Sources

Keith, Mark. (Oct 11, 2021). Python: MLR, OLS, standardization, normalization. YouTube. Retrieved from https://www.youtube.com/watch?v=QH_elD_JKuc&list=PLe9UEU4oeAuV7RtCbL76hca5ELO_IELk4&index=11

Sewell, William. (April 2023). D208 Predictive Modeling Webinar Episode 1. Panopto. Retrieved from https://westerngovernorsuniversity-my.sharepoint.com/:p:/g/personal/william_sewell_wgu_edu/ER_vJMbYtxJGpxImpZ0DUQcBoVcORYKanFVKNKFcEXkRow?rtime=03cpw43G2kg

I. Additional Sources

Mahmood, M.S. (April 2021). Simple explanation of statsmodel linear regression model summary. Towards Data Science. https://towardsdatascience.com/simple-explanation-of-statsmodel-linear-regression-model-summary-35961919868b

Middleton, Keiona. (October 2022). D208 - Webinar: Getting Started with D208 Part II [Video]. Panopto. Retrieved from https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=09b8fdbb-a374-452b-ba53-af39001ff3f3

Middleton, Keiona. (November 2022). D208 - Webinar: Getting Started with D208 Part I [Video]. Panopto. Retrieved from https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=15e09c73-c5aa-439d-852f-af47001b8970

pandas. (2024). Returning a view versus a copy – Use of .iloc. In pandas documentation. Retrieved from https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

Reed, A. (n.d.). Machine Learning - Multi Linear Regression Analysis. GitHub. Retrieved June 6, 2024, from https://github.com/areed1192/sigma_coding_youtube/blob/master/python/python-data-science/machine-learning/multi-linear-regression/Machine%20Learning%20-%20Multi%20Linear%20Regression%20Analysis.ipynb

Tripathi, Ashutosh. (Jun 10, 2019). "Feature Selection Techniques in Regression Model". Towards Data Science. Retrieved from https://towardsdatascience.com/feature-selection-techniques-in-regression-model-26878fe0e24e

Western Governors University. (2024). R or Python: Choosing a Language for Data Science. Retrieved from https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html