|  |
| --- |
| Modeling Length of Inpatient Admissions |
| Western Governors University |
| Heather Rauch |

Table of Contents

[A: Research Question 2](#_Toc116064431)

[A1: Research Question 2](#_Toc116064432)

[A2: Objectives and Goals 2](#_Toc116064433)

[B: Method Justification 2](#_Toc116064434)

[B1: Summary of Assumptions 2](#_Toc116064435)

[B2: Tool Benefits 2](#_Toc116064436)

[B3: Appropriate Technique 3](#_Toc116064437)

[C: Data Preparations 3](#_Toc116064438)

[C1: Data Goals 3](#_Toc116064439)

[C2: Summary Statistics 3](#_Toc116064440)

[C3: Steps to Prepare the Data 5](#_Toc116064441)

[C4: Visualizations 10](#_Toc116064442)

[C5: Prepared Data Set 10](#_Toc116064443)

[D: Model Comparison 11](#_Toc116064444)

[D1: Initial Model 11](#_Toc116064445)

[D2: Justification of Model Reduction 11](#_Toc116064446)

[D3: Reduced Multiple Regression Model 12](#_Toc116064447)

[E: Model Analysis 13](#_Toc116064448)

[E1: Model Comparison 13](#_Toc116064449)

[E2: Output and Calculations 13](#_Toc116064450)

[E3: Code 18](#_Toc116064451)

[F: Data Summary and Implications 20](#_Toc116064452)

[F1: Results 20](#_Toc116064453)

[F2: Recommendations 21](#_Toc116064454)

[G: Panopto Recording 22](#_Toc116064455)

[H: Sources of Third-Party Code 22](#_Toc116064456)

[I: Sources 22](#_Toc116064457)

# A: Research Question

## A1: Research Question

This analysis aims to answer the question: What factors influence the length of inpatient hospital admissions?

## A2: Objectives and Goals

The primary goal and supporting objectives of this data analysis are as follows:

* Goal: Gain a better understanding of inpatient hospital admission length.
  + Objective 1: Determine which features have a statistically significant impact on inpatient hospital admission length.
  + Objective 2: Develop a linear model of inpatient hospital admission length using the identified features.
  + Objective 3: Identify real-world scenarios which could reduce inpatient hospital admission length using the model.

# B: Method Justification

## B1: Summary of Assumptions

There are five assumptions of multiple regression modeling (Bobbitt, 2021):

1. Each independent variable has a linear relationship with the dependent variable.
2. There is no multicollinearity between the independent variables.
3. Each observation in the data set is independent.
4. The distribution of the residuals is normal.
5. The residuals’ variance is maintained across the model.

## B2: Tool Benefits

To complete all phases of the analysis Python was used. For data cleaning, the numpy and pandas libraries allow Structured Query Language style operations. These make data cleaning far easier than using a tool like spreadsheet software, which can be problematic with large data sets.

For the more advanced operations like data wrangling, and the analysis itself python has available statistical libraries like scipy, statsmodels, and scikit-learn. Between the three of them, just about any analysis is not only possible, but simple to implement. Just like with the data cleaning process, python is superior to spreadsheet software. To perform a regression analysis in such software, many intermediate steps must be performed. This is contrary to the one-line statements available in the python libraries.

To produce visualizations, python was also used with the matplotlib and seaborn libraries. There are several benefits to this, with the top being simplifying the process. Having already completed the other two portions using python, it was simpler to use python for everything. The data was readily available and matplotlib is an excellent visualization library.

## B3: Appropriate Technique

The question being researched aims to discover which factors contribute most to the length of inpatient admissions. Regression analysis is used to understand the relationship between a single response variable and many predictor variables. More specifically, multiple linear regression analysis is designed to handle a response variable that is numeric rather than categorical. Because the response variable, Initial\_days, is not only numeric but continuous, multiple linear regression is the most practical analysis for the question being asked.

# C: Data Preparations

## C1: Data Goals

The goals of the data preparation process were to clean, explore, and wrangle the data in preparation for multiple linear regression analysis.

During the cleaning phase, the data set will be checked for duplicate observations, missing values, and outliers. If any duplicate observations are identified, they will be removed. If any missing values are identified, they will either be removed or imputed. If any outliers are identified, they will be investigated individually to determine if any further action is required.

During the exploration phase, univariate and bivariate analyses will be performed. Each variable will be assessed individually to understand its respective distribution. Then, each predictor variable will be assessed with the response variable to gain a better understanding of the individual relationships.

During the last phase, data wrangling, three activities will take place. First, all categorical variables will be re-coded to numeric either by mapping values to the categories or by creating dummy variables to represent the categories. The second activity will be scaling the numeric variables. This will ensure the magnitude of each variable does not skew the model. The last step in the data wrangling process will be to check the variance influence factors of each of the variables. This is to ensure that no multicollinearity exists within the dataset. If any variables are found to have a VIF of 10 or more, one will be removed to reduce their effect on the model.

## C2: Summary Statistics

Among the quantitative variables, Income has the widest range while vitD\_supp has the smallest range. TotalCharge and Additional\_charges are also variables with quite large numeric values. These will need to be scaled during the data wrangling phase to prevent them from skewing the regression analysis.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable** | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| Initial\_days | 10000 | 34.4553 | 26.3093 | 1.0020 | 7.8962 | 35.8362 | 61.1610 | 71.9815 |
| Lat | 10000 | 38.7511 | 5.4031 | 17.9672 | 35.2551 | 39.4194 | 42.0442 | 70.5610 |
| Lng | 10000 | -91.2431 | 15.2060 | -174.2097 | -97.3530 | -88.3972 | -80.4381 | -65.2902 |
| Children | 10000 | 2.0972 | 2.1637 | 0.0000 | 0.0000 | 1.0000 | 3.0000 | 10.0000 |
| Age | 10000 | 53.5117 | 20.6385 | 18.0000 | 36.0000 | 53.0000 | 71.0000 | 89.0000 |
| Income | 10000 | 40490.4952 | 28521.1533 | 154.0800 | 19598.7750 | 33768.4200 | 54296.4025 | 207249.1000 |
| VitD\_levels | 10000 | 17.9643 | 2.0172 | 9.8065 | 16.6264 | 17.9511 | 19.3480 | 26.3944 |
| Doc\_visits | 10000 | 5.0122 | 1.0457 | 1.0000 | 4.0000 | 5.0000 | 6.0000 | 9.0000 |
| Full\_meals\_eaten | 10000 | 1.0014 | 1.0081 | 0.0000 | 0.0000 | 1.0000 | 2.0000 | 7.0000 |
| vitD\_supp | 10000 | 0.3989 | 0.6285 | 0.0000 | 0.0000 | 0.0000 | 1.0000 | 5.0000 |
| TotalCharge | 10000 | 5312.1728 | 2180.3938 | 1938.3121 | 3179.3740 | 5213.9520 | 7459.6998 | 9180.7280 |
| Additional\_charges | 10000 | 12934.5286 | 6542.6015 | 3125.7030 | 7986.4878 | 11573.9777 | 15626.4900 | 30566.0700 |

The qualitative or categorical variables are largely made up of yes and no values. These will lend themselves well to encoding during the wrangling phase. Area, Marital, Gender, Initial\_admin, Complication\_risk, and Services all have three or more levels which will require different handling than the yes and no values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable** | **count** | **unique** | **top** | **freq** | **categories** |
| Soft\_drink | 10000 | 2 | No | 7425 | No, Yes |
| HighBlood | 10000 | 2 | No | 5910 | Yes, No |
| Stroke | 10000 | 2 | No | 8007 | No, Yes |
| Overweight | 10000 | 2 | Yes | 7094 | No, Yes |
| Arthritis | 10000 | 2 | No | 6426 | Yes, No |
| Diabetes | 10000 | 2 | No | 7262 | Yes, No |
| Hyperlipidemia | 10000 | 2 | No | 6628 | No, Yes |
| BackPain | 10000 | 2 | No | 5886 | Yes, No |
| Anxiety | 10000 | 2 | No | 6785 | Yes, No |
| Allergic\_rhinitis | 10000 | 2 | No | 6059 | Yes, No |
| Reflux\_esophagitis | 10000 | 2 | No | 5865 | No, Yes |
| Asthma | 10000 | 2 | No | 7107 | Yes, No |
| Area | 10000 | 3 | Rural | 3369 | Suburban, Urban, Rural |
| Marital | 10000 | 5 | Widowed | 2045 | Divorced, Married, Widowed, Never Married, Separated |
| Gender | 10000 | 3 | Female | 5018 | Male, Female, Nonbinary |
| Initial\_admin | 10000 | 3 | Emergency Admission | 5060 | Emergency Admission, Elective Admission, Observation Admission |
| Complication\_risk | 10000 | 3 | Medium | 4517 | Medium, High, Low |
| Services | 10000 | 4 | Blood Work | 5265 | Blood Work, Intravenous, CT Scan, MRI |

## C3: Steps to Prepare the Data

The python code provided below was used to prepare the medical data for final analysis. This and all other code used for the analysis can be found attached to the submission as task1.py.

To configure the coding environment for the rest of the analysis, several libraries were imported, an output directory was created, and the data set was loaded into a pandas DataFrame.

import os  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from scipy import stats  
from sklearn.preprocessing import StandardScaler  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
from statsmodels.formula.api import ols  
  
os.makedirs('./output', exist\_ok=True)

medical\_clean = pd.read\_csv(r"C:\Users\user\OneDrive\Documents\Education\Western Govenors University"  
 r"\MS - Data Analytics\D208 - Predictive Modeling\medical\_clean.csv")

Next, the dependent variable, Initial\_days, and the independent variables were selected. The independent variables included Lat, Lng, Children, Age, Income, VitD\_levels, Doc\_vsits, Full\_meals\_eaten, vitD\_supp, TotalCharge, Additional\_charges, Soft\_drink, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic\_rhinitis, Reflux\_esophagitis, Asthma, Area, Marital, Gender, Initial\_admin, Complication\_risk, and Services.

Once the variables of interest were selected, the data was cleaned. This was accomplished by checking for duplicate observations, missing values, and outliers.

columns = ['Initial\_days', 'Lat', 'Lng', 'Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten',  
 'vitD\_supp', 'TotalCharge', 'Additional\_charges', 'Soft\_drink', 'HighBlood', 'Stroke', 'Overweight',  
 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis',  
 'Asthma', 'Area', 'Marital', 'Gender', 'Initial\_admin', 'Complication\_risk', 'Services']  
df = medical\_clean[columns]

There were no duplicate observations or missing values. However, there were several variables with outliers. Each outlier, or set of outliers where it made sense, was investigated to ensure the data contained no errors. The Lat and Lng outliers were reviewed together since they were somewhat meaningless without one another.

location\_outliers = medical\_clean.loc[df\_outliers.Lat | df\_outliers.Lng, ['State', 'Lat', 'Lng']]  
location\_outliers.columns = ['State', 'Lat\_mean', 'Lng\_mean']  
location\_outlier\_groups = location\_outliers.groupby("State", as\_index=False).mean()  
print(f'Location Outliers:\n{location\_outlier\_groups.to\_string()}')

print(f'Duplication check:\n\tNumber of duplicated rows: {df.duplicated().sum()}')  
print(f'Null detection:\n{df.isna().sum()}')  
  
num\_cols = [column for column in columns if df[column].dtype in ['float64', 'int64']]  
df\_num = df[num\_cols]  
df\_zscores = df\_num.apply(stats.zscore)  
df\_outliers = df\_zscores.apply(lambda x: (x > 3) | (x < -3))  
print(f'Outlier counts:\n{df\_outliers.sum().to\_string()}')

When the Lat and Lng outliers were grouped by State, it became apparent that the outliers represented Alaska, Hawaii, and Puerto Rico. All three of these represent locations not part of the continental United States. Since they are still part of the United States as a whole, they were not excluded from the final analysis.

The Children outliers were investigated next.

children\_outliers = medical\_clean.loc[df\_outliers.Children, ['Children', 'Customer\_id']]  
children\_outliers.columns = ['Children', 'count']  
children\_outliers = children\_outliers.groupby("Children", as\_index=False).count()  
print(f'Children Outliers:\n{children\_outliers.to\_string()}')

While having nine or 10 children was certainly not the norm in this data, the values did not appear to be erroneous. For that reason, the observations were not excluded.

Next, the Income outliers were reviewed along with the Job variable for context.

income\_outliers = medical\_clean.loc[df\_outliers.Income, ['Job', 'Income']]  
income\_outliers.columns = ['Job', 'Income\_mean']  
income\_outliers = income\_outliers.groupby("Job", as\_index=False).mean(). \  
 sort\_values("Income\_mean", ascending=False)  
print(f'Income Outliers:\n{income\_outliers.iloc[:10].to\_string()}')

Most of the job titles were those known to receive well above the average compensation (e.g., Chief Executive Officer, Software engineer, etc.). It would not make sense to exclude completely legitimate income values.

The VitD\_levels and vitD\_supp outliers were investigated together since vitamin D supplements get prescribed for vitamin D deficiency.

vitd\_outliers = df.loc[:, ['VitD\_levels', 'vitD\_supp']]  
vitd\_outliers['VitD\_levels\_outlier'] = df\_outliers.VitD\_levels  
vitd\_outliers['vitD\_supp\_outlier'] = df\_outliers.vitD\_supp  
vitd\_outliers = vitd\_outliers.groupby(['VitD\_levels\_outlier', 'vitD\_supp\_outlier'], as\_index=False). \  
 mean().sort\_values(['VitD\_levels\_outlier', 'vitD\_supp\_outlier'], ascending=False)  
print(f'Vitamin D Outliers\n{vitd\_outliers.to\_string()}')

In both the VitD\_levels and vitD\_supp outliers, the mean vitamin D levels and mean vitamin D supplements were above the mean for those who were not outliers for either variable. Since the goal of this analysis was to determine what impacts the length of inpatient admissions, it would not be logical to exclude variables directly relevant to the patient's health.

The Doc\_visits outliers were investigated next.

doc\_visits\_outliers = medical\_clean.loc[df\_outliers.Doc\_visits, ['Doc\_visits', 'Customer\_id']]  
doc\_visits\_outliers.columns = ['Doc\_visits', 'count']  
doc\_visits\_outliers = doc\_visits\_outliers.groupby('Doc\_visits', as\_index=False).count()  
print(f'Doc\_visits Outliers:\n{doc\_visits\_outliers.to\_string()}')

There were only eight outliers in the Doc\_visits variable, to begin with. The outliers were one and nine doctor visits. Depending on how long the patients were hospitalized, both could be completely valid. None of them were excluded.

The last set of outliers investigated was Full\_meals\_eaten.

full\_meals\_eaten\_outliers = medical\_clean.loc[df\_outliers.Full\_meals\_eaten, ['Full\_meals\_eaten', 'Customer\_id']]  
full\_meals\_eaten\_outliers.columns = ['Full\_meals\_eaten', 'count']  
full\_meals\_eaten\_outliers = full\_meals\_eaten\_outliers.groupby('Full\_meals\_eaten', as\_index=False).count()  
print(f'Full\_meals\_eaten Outliers:\n{full\_meals\_eaten\_outliers.to\_string()}')

The values for the Full\_meals\_eaten outliers ranged from five to seven with almost three-quarters of them having a value of five. Considering the length of some of the stays in this data set, it was surprising that the values were not higher. A total of seven full meals seemed low for someone hospitalized for 60 days. Without knowing the medical conditions, for instance, whether the patient was conscience during their stay, it is not possible to determine whether the outliers were erroneous. To not unintentionally skew the data, the outliers were not excluded.

Once the data cleaning process was complete, exploratory data analysis was performed. Each of the selected variables in the cleaned data was plotted to view their distributions.

s, e = 0, 6  
grid = []  
for i in range(int(np.ceil(df.shape[1] / e))):  
 grid.append([x for x in df.columns.values[s:e]])  
 s = e  
 e += 6  
fig, ax = plt.subplots(len(grid), 6, figsize=(15, 8))  
fig.set\_tight\_layout(True)  
fig.suptitle('Univariate Analysis')  
for r, row in enumerate(grid):  
 for c, col in enumerate(row):  
 if df[col].dtype in ('float64', 'int64'):  
 ax[r, c].hist(df[col])  
 ax[r, c].vlines(df[col].median(), 0, 4000, color='black', label='median')  
 ax[r, c].vlines(df[col].mean(), 0, 4000, color='red', linestyles='--', label='mean')  
 else:  
 df\_grp = df.groupby(col).count()  
 if col == 'Initial\_admin':  
 df\_grp.index = df\_grp.index.str.replace(' Admission', '')  
 ax[r, c].bar(df\_grp.index, df\_grp.values[:, 0])  
 plt.setp(ax[r, c].get\_xticklabels(), rotation=20, ha='right', rotation\_mode='anchor')  
 ax[r, c].set\_xlabel(col)  
plt.savefig('./output/univariate\_analysis.png')  
plt.show()

All of the variables were displayed in a single figure, each having its subplot. Quantitative variables were displayed as histograms with solid black lines marking the median and dashed red lines marking the mean. The addition of median and mean lines allowed the assessment of skew in each variable. Qualitative variables were displayed as bar charts.

Following the univariate analysis, each independent variable was plotted against the dependent variable, Initial\_days, to assess their relationships.

s, e = 0, 6  
grid = []  
for i in range(int(np.ceil(len(df.columns) / e))):  
 grid.append([x for x in df.columns[1:].values[s:e]])  
 s = e  
 e += 6  
fig, ax = plt.subplots(len(grid), 6, sharey='all', figsize=(15, 8))  
fig.set\_tight\_layout(True)  
fig.suptitle('Bivariate Analysis')  
for r, row in enumerate(grid):  
 for c, col in enumerate(row):  
 if df[col].dtype in ('float64', 'int64'):  
 ax[r, c].scatter(df[col], df.Initial\_days, marker='.', alpha=.25)  
 else:  
 df\_grp = {}  
 for grp in df.loc[:, col].unique():  
 df\_grp[grp] = df.loc[df[col].eq(grp), 'Initial\_days'].values  
 labels = [x.replace(' Admission', '') for x in df\_grp.keys()]  
 ax[r, c].boxplot(df\_grp.values(), labels=labels, widths=0.5, patch\_artist=True,  
 medianprops={'color': 'white', 'linewidth': 0.5},  
 boxprops={'facecolor': 'C0', 'edgecolor': 'white', 'linewidth': 0.5},  
 whiskerprops={'color': 'C0', 'linewidth': 1.5},  
 capprops={'color': 'C0', 'linewidth': 1.5})  
 plt.setp(ax[r, c].get\_xticklabels(), rotation=20, ha='right', rotation\_mode='anchor')  
 ax[r, c].set\_xlabel(col)  
 if c == 0:  
 ax[r, c].set\_ylabel('Initial\_days')  
plt.savefig('./output/bivariate\_analysis.png')  
plt.show()

Once again, all of the plots were displayed in a single figure using subplots. Quantitative independent variables were plotted against Initial\_days using scatter plots. Qualitative independent variables were plotted against Initial\_days using box plots.

Next, the data wrangling process took place. This consisted of recoding categorical variables, scaling numeric variables, and removing variables causing multicollinearity in the data set.

During the re-coding process, Yes and No values were replaced with 1 and 0, respectively. The remaining categorical variables were replaced with dummy variables. One new variable was produced for every category minus one to prevent multicollinearity in the data. The spaces in the newly created column names were also replaced with underscores to allow a more programmatic approach during the model reduction process.

codes = {'Yes': 1, 'No': 0}  
df = df.replace(codes)

df = pd.get\_dummies(df, columns=['Area', 'Marital', 'Gender', 'Initial\_admin', 'Services', 'Complication\_risk'],  
 drop\_first=True)

columns = [col.replace(' ', '\_') for col in df.columns]  
df.columns = columns

print(f'Recoded data:\n{df.head().to\_string()}')

After re-coding the categorical variables, the numeric variables were scaled to prevent the magnitude of each variable from skewing the analysis.

The last step before building the model was checking for multicollinearity since it was one of the assumptions of multiple linear regression.

x = df.loc[:, df.columns[1:]]  
vif\_data = pd.DataFrame()  
vif\_data['feature'] = x.columns  
vif\_data['vif'] = [variance\_inflation\_factor(x.values, i) for i in range(x.shape[1])]  
print(f'Variance Influence Factor:\n{vif\_data.to\_string()}')

scaler = StandardScaler()  
df.loc[:, df\_num.columns] = scaler.fit\_transform(df\_num)  
print(f'Scaled Data:\n{df.head().to\_string()}')

To check for multicollinearity, the variance influence factor for each variable was assessed. The Additional\_charges variable had a VIF of 13.21 and HighBlood 10.49. The Additional\_charges variable was removed and the VIF was reevaluated.

df.drop('Additional\_charges', axis=1, inplace=True)  
x = df.loc[:, df.columns[1:]]  
vif\_data = pd.DataFrame()  
vif\_data['feature'] = x.columns  
vif\_data['vif'] = [variance\_inflation\_factor(x.values, i) for i in range(x.shape[1])]  
print(f'Updated Variance Influence Factor:\n{vif\_data.to\_string()}')

The new VIF values were all well under the threshold of ten.

## C4: Visualizations

Because all of the variables were visualized in a single plot, making them difficult to view as figures in a document such as this, they were also included as attachments with the task submission. The attached files are titled univariate\_analysis.png and bivariate\_analysis.png.

Diagram, timeline

Description automatically generated

Graphical user interface, diagram

Description automatically generated

## C5: Prepared Data Set

The prepared dataset has been included with the task submission as prepared\_data\_set.csv.

# D: Model Comparison

## D1: Initial Model

Table

Description automatically generated

## D2: Justification of Model Reduction

Variable selection will be performed recursively using each variable’s p-value. One at a time, the variable with the highest p-value will be removed until none above .05 remains. This will ensure that only variables whose impact on Initial\_days is statistically significant will remain. Overall model evaluation will be assessed using adjusted r2, the f-statistic, and the f-statistic probability or p-value.

## D3: Reduced Multiple Regression Model

Table

Description automatically generated

# E: Model Analysis

## E1: Model Comparison

Variable selection was performed using a recursive user-defined function. The function looked at the p-values for each independent variable and removed the variable with the highest p-value. It then rebuilt the model, fit the model, and looked at the new p-values. If any of them were over 0.05, the function would call itself repeatedly on the new model until no more insignificant independent variables remained. Once the model was reduced, the gross and reduced models were compared using the adjusted r2, the f-statistic, the f-statistic p-value, the residual standard error, as well as a visual inspection of the residual distributions shown below. Referencing the output from each model, the adjusted r2 and f-statistic probability did not change. However, the f-statistic more than tripled. This means that the reduced model, as a whole, is more meaningful than the gross model.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Gross Model** | **Reduced Model** |
| Adjusted r2 | 1.000 | 1.000 |
| F-statistic | 1.829e+16 | 5.645e+16 |
| Prob (F-statistic) | 0.00 | 0.00 |

Chart, histogram

Description automatically generated

## E2: Output and Calculations

Removing Overweight with a p-value of 0.9868967505885853

Removing Gender\_Male with a p-value of 0.9771056309884012

Removing Soft\_drink with a p-value of 0.938105737820849

Removing Initial\_admin\_Observation\_Admission with a p-value of 0.8532182454921745

Removing vitD\_supp with a p-value of 0.8066251852204682

Removing Lng with a p-value of 0.7603706511218857

Removing Income with a p-value of 0.7545580338784263

Removing Services\_CT\_Scan with a p-value of 0.7312704069497018

Removing Stroke with a p-value of 0.7267418652089916

Removing Lat with a p-value of 0.7235515162887551

Removing Gender\_Nonbinary with a p-value of 0.7236477315696986

Removing Doc\_visits with a p-value of 0.6420438628685536

Removing Asthma with a p-value of 0.4772458205026635

Removing Services\_Intravenous with a p-value of 0.46549594310387066

Removing Services\_MRI with a p-value of 0.4839185369333112

Removing Age with a p-value of 0.4322265745367786

Removing Children with a p-value of 0.3722677452998736

Removing Marital\_Separated with a p-value of 0.36970431306469487

Removing Marital\_Widowed with a p-value of 0.35966448704853893

Removing Marital\_Married with a p-value of 0.48680030534818963

Removing VitD\_levels with a p-value of 0.33727003674647704

Removing Marital\_Never\_Married with a p-value of 0.23723655290361637

Removing Full\_meals\_eaten with a p-value of 0.1763645427157862

Removing Area\_Urban with a p-value of 0.10672578161713468

Removing Area\_Suburban with a p-value of 0.13856709803628658

Residual Standard Error:

Gross Model: 1.2155608651737813e-07

Reduced Model: 1.2149515978440607e-07

Timeline

Description automatically generated

Variance Influence Factor:

feature vif

0 TotalCharge 1.022292

1 HighBlood 1.557312

2 Arthritis 1.470733

3 Diabetes 1.315923

4 Hyperlipidemia 1.422500

5 BackPain 1.567742

6 Anxiety 1.399028

7 Allergic\_rhinitis 1.529129

8 Reflux\_esophagitis 1.566865

9 Initial\_admin\_Emergency\_Admission 1.806890

10 Complication\_risk\_Low 1.428633

11 Complication\_risk\_Medium 1.909267

Chart, scatter chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Predictions:

count mean std min 25% 50% 75% max

TotalCharge 36864.0 5312.172769 2180.314388 1740.089964 3421.070108 5312.172769 7203.275430 8884.255573

HighBlood 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Arthritis 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Diabetes 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Hyperlipidemia 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

BackPain 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Anxiety 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Allergic\_rhinitis 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Reflux\_esophagitis 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Initial\_admin\_Emergency\_Admission 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Complication\_risk\_Low 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Complication\_risk\_Medium 36864.0 0.500000 0.500007 0.000000 0.000000 0.500000 1.000000 1.000000

Initial\_days 36864.0 35.122121 27.066284 -20.581484 12.042997 35.122121 58.201245 90.825726

Chart

Description automatically generated

## E3: Code

*# Gross Linear Model*

columns = df.columns.values.tolist()  
columns.pop(0)  
formula = f'Initial\_days ~ {" + ".join([col for col in columns])}'  
model1 = ols(formula, data=df).fit()  
print(model1.summary())

*# Feature Selection*

def reduce\_model(model: ols, data: pd.DataFrame) -> ols:  
 features = model.params.index.to\_list() *# Get the list of features* features.pop(0) *# Remove the intercept from the list of features* highest\_p = model.pvalues.argmax() - 1 *# Get the index of the highest p-value* print(f'Removing {features[highest\_p]} with a p-value of {float(model.pvalues[highest\_p + 1])}')  
 features.pop(highest\_p) *# Remove the item with the highest p-value* formula = f'Initial\_days ~ {" + ".join(features)}' *# Rewrite the formula* model = ols(formula=formula, data=data).fit() *# Rebuild the model* if any(model.pvalues.round(2)) > 0.05: *# Recursively reduce the model until no insignificant features remain* model = reduce\_model(model, data)  
 return model  
  
*# Reduced Linear Model*

model2 = reduce\_model(model1, df)

print(model2.summary())

*# Residuals Gross vs Reduced*  
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4))  
fig.suptitle('Residuals Distributions')  
ax1.set\_title('Gross Model')  
ax1.hist(model1.resid)  
ax2.set\_title('Reduced Model')  
ax2.hist(model2.resid)  
plt.show()

*# Calculate residual standard error from Mean Squared Error (mse)*print(f'Residual Standard Error:\n'  
 f'\tGross Model: {np.sqrt(model1.mse\_resid)}\n'  
 f'\tReduced Model: {np.sqrt(model2.mse\_resid)}\n')

df\_model = df.loc[:, ['Initial\_days', 'TotalCharge', 'HighBlood', 'Arthritis', 'Diabetes', 'Hyperlipidemia',  
 'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis',  
 'Initial\_admin\_Emergency\_Admission', 'Complication\_risk\_Low', 'Complication\_risk\_Medium']]  
corr = df\_model.corr()  
fig, ax = plt.subplots(figsize=(8, 6))  
fig.set\_tight\_layout(True)  
fig.suptitle('Correlation Matrix')  
im = ax.imshow(corr, vmin=-1, vmax=1, cmap='Spectral')  
cbar = ax.figure.colorbar(im, ax=ax)  
ax.set\_xticks(np.arange(corr.shape[1]), labels=corr.columns)  
ax.set\_yticks(np.arange(corr.shape[0]), labels=corr.index)  
plt.setp(ax.get\_xticklabels(), rotation=25, ha='right', rotation\_mode='anchor')  
for i, index in enumerate(corr.index):  
 for c, col in enumerate(corr.columns):  
 text = ax.text(c, i, '{:.4f}'.format(corr.loc[index, col]).replace('0.', '.').  
 replace('1.0000', ''), ha='center', va='center', size=7,  
 color='w' if corr.loc[index, col] > 0.5 else 'black')  
plt.show()

*# Check Multicollinearity*x = df\_model.loc[:, df\_model.columns[1:]]  
vif\_data = pd.DataFrame()  
vif\_data['feature'] = x.columns  
vif\_data['vif'] = [variance\_inflation\_factor(x.values, i) for i in range(x.shape[1])]  
print(f'Variance Influence Factor:\n{vif\_data.to\_string()}')

*# Check Independence of Observations and Homosceasticity*

sns.regplot(x=model2.fittedvalues, y=model2.resid, lowess=True, line\_kws={'color': 'red'})  
plt.title('Residuals vs. fitted')  
plt.xlabel('Fitted values')  
plt.ylabel('Residuals')  
plt.show()

*# Check Normality of residuals*

qqplot(data=model2.resid, fit=True, line='45')  
plt.title('Q-Q plot')  
plt.show()

*# Scale-location plot*

model2\_norm\_resid = model2.get\_influence().resid\_studentized\_internal  
model2\_norm\_resid\_abs\_sqrt = np.sqrt(np.abs(model2\_norm\_resid))  
sns.regplot(x=model2.fittedvalues, y=model2\_norm\_resid\_abs\_sqrt, ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-location plot')  
plt.xlabel('Fitted values')  
plt.ylabel('|Standardized Residuals|\*\*(1/2)')  
plt.show()

*# Create prediction data*total\_charge = np.arange(2000, 11000, 500)  
high\_blood = np.array([0, 1], int)  
arthritis = np.array([0, 1], int)  
diabetes = np.array([0, 1], int)  
hyperlipidemia = np.array([0, 1], int)  
back\_pain = np.array([0, 1], int)  
anxiety = np.array([0, 1], int)  
allergic\_rhinitis = np.array([0, 1], int)  
reflux\_esophagitis = np.array([0, 1], int)  
emergency\_admission = np.array([0, 1], int)  
low\_risk = np.array([0, 1], int)  
medium\_risk = np.array([0, 1], int)  
p = product(total\_charge, high\_blood, arthritis, diabetes, hyperlipidemia, back\_pain, anxiety, allergic\_rhinitis,  
 reflux\_esophagitis, emergency\_admission, low\_risk, medium\_risk)  
explanatory\_data = pd.DataFrame(p, columns=df\_model.columns[1:])

*# Scale numeric part of data*scaler = StandardScaler()  
scaler.fit(explanatory\_data.loc[:, ['TotalCharge', 'HighBlood']])  
explanatory\_data.loc[:, ['TotalCharge', 'HighBlood\_']] = \  
 scaler.transform(explanatory\_data.loc[:, ['TotalCharge', 'HighBlood']])  
explanatory\_data.drop('HighBlood\_', axis=1, inplace=True)

*# Make predictions*prediction\_data = explanatory\_data.assign(Initial\_days=model2.predict(explanatory\_data))

*# Unscale data for interpretation*scaler = StandardScaler()  
scaler.fit(medical\_clean.loc[:, ['TotalCharge', 'Initial\_days']])  
prediction\_data.loc[:, ['TotalCharge', 'Initial\_days']] = \  
 scaler.inverse\_transform(prediction\_data.loc[:, ['TotalCharge', 'Initial\_days']])  
df.loc[:, ['TotalCharge', 'Initial\_days']] = scaler.inverse\_transform(df.loc[:, ['TotalCharge', 'Initial\_days']])  
print(f'Predictions:\n{prediction\_data.describe().T.to\_string()}')

*# Plot predictions*sns.scatterplot(x=df.TotalCharge, y=df.Initial\_days, hue=df.Initial\_admin\_Emergency\_Admission, alpha=.25)  
sns.scatterplot(x=prediction\_data.TotalCharge, y=prediction\_data.Initial\_days,  
 hue=prediction\_data.Initial\_admin\_Emergency\_Admission, legend=False, marker='s', alpha=.25)  
plt.title('Predicted vs. Actual')  
plt.show()

# F: Data Summary and Implications

## F1: Results

The regression equation for the reduced model is:

Each coefficient can be interpreted as follows:

* **TotalCharge:** With all other variables held constant, a one-unit increase in TotalCharge is associated with a 1.0114 unit increase in Initial\_days.
* **HighBlood:** With all other variables held constant, having high blood pressure is associated with a 5.21% shorter inpatient admission on average than those who do not have high blood pressure.
* **Arthritis:** With all other variables held constant, having arthritis is associated with a 3.34% shorter inpatient admission on average than those who do not have arthritis.
* **Diabetes:** With all other variables held constant, having diabetes is associated with a 3.49% shorter inpatient admission on average than those who do not have diabetes.
* **Hyperlipidemia:** With all other variables held constant, having hyperlipidemia is associated with a 4.36% shorter inpatient admission on average than those who do not have hyperlipidemia.
* **BackPain:** With all other variables held constant, having back pain is associated with a 3.95% shorter inpatient admission on average than those who do not have back pain.
* **Anxiety:** With all other variables held constant, having anxiety is associated with a 3.99% shorter inpatient admission than those who do not have anxiety.
* **Allergic\_rhinitis:** With all other variables held constant, having allergic rhinitis is associated with a 2.81% shorter inpatient admission than those who do not have allergic rhinitis.
* **Reflux\_esophagitis:** With all other variables held constant, having reflux esophagitis is associated with a 2.77% shorter inpatient admission than those who do not have reflux esophagitis.
* **Initial\_admin\_Emergency\_Admission:** With all other variables held constant, being admitted from the Emergency Department is associated with a 23.77% shorter inpatient admission than those who were admitted as for an elective procedure or observation.
* **Complication\_risk\_Low:** With all other variables held constant, having a low risk of complications is associated with a 19.18% longer inpatient admission than those who have a high risk of complications.
* **Complication\_risk\_Medium:** With all other variables held constant, having a medium risk of complications is associated with a 19.18% longer inpatient admission than those who have a high risk of complications.

The reduced model is statistically significant. During the model comparison, it was revealed that the adjusted r2 value was 1.0, meaning the independent variables explained 100% of the variance in the length of initial days. The f-statistic was 5.645e+16 which is very large. The associated p-value was 0.0 meaning these findings did not just happen by chance. Looking at each independent variable individually, the p-values were all 0.000, indicating that each of them was also statistically significant and not just contributing by chance.

The reduced model is also practically significant in some contexts. It predicts, with excellent accuracy, how long a patient’s initial admission will be given the inpatient charges, whether the patient was admitted from the Emergency Department, the patient’s complication risk, and whether the patient has high blood pressure, arthritis, diabetes, hyperlipidemia, back pain, anxiety, allergies, or acid reflux; but only for stays under 30 days in length. For stays over 30 days, the variance in the model’s residuals increases, making the predictions less reliable. This, however, does not make the model useless. It can certainly predict short stays reliably, and it can predict when a stay will be over one month. This is valuable information to the hospital administration because it helps to inform their census for occupancy planning.

The data analysis did have a few limitations, starting with the data cleaning process. During the outlier investigation, all outliers were retained. Regression analysis is not resilient to outliers, which can weaken the model’s ability to extrapolate beyond the sample. Another limitation was introduced during the data wrangling phase by scaling the data. Any transformations to the data make it more complex to interpret. This also means that any data fed into the model for practical use will also need to be scaled for predictions and unscaled for interpretation. The third limitation is that no causation can be implied from the correlations observed. There are clear patterns present that indicate there *could be* causation. However, that can’t be confirmed without performing a properly controlled experiment. The last and most significant limitation of this analysis is that hospital charges are incurred as the length of stay increases. This means that, unless a large number of charges are incurred at the beginning of a patient's stay, for instance, due to a surgical procedure, the patient’s stay will never look like it will be longer than it already is.

## F2: Recommendations

Based on these findings, there are a few courses of action that should be taken. First, at least one controlled experiment should be designed and implemented to determine which, if any, of the independent variables cause or contribute to an increase in the length of inpatient admissions. Likely lurking variables are interacting with the model which could better explain the variation in longer admissions. For that reason, the second course of action which should be taken is further analysis. Some indication of the state of the patient's health, like admitting diagnosis or chief complaint, as well as whether the patients had any surgical procedures during their stay would be great information to include. The final course of action which should be taken is to use the actionable information the model does provide. Previously, it was shown that the model performs well on admissions up to 30 days in length. This information should be used by the hospital administration to inform their census and occupancy planning. If they know how many and which rooms of the hospital may still be occupied for an extended length of time they can determine in advance if additional measures need to be taken to accommodate incoming patients. This would give them additional time to implement these measures.

# G: Panopto Recording

[The Panopto recording can be accessed at the following address: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=506470ba-068c-4392-86b0-af2701704a9b](The%20Panopto%20recording%20can%20be%20accessed%20at%20the%20following%20address:%20https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=506470ba-068c-4392-86b0-af2701704a9b)

# H: Sources of Third-Party Code

Only library reference documentation was used to produce the included code. This includes the following Python libraries: NumPy, pandas, Matplotlib, Seaborn, Scikit-learn, and Statsmodels.

# I: Sources

Bobbitt, Z. (2021, November 16). *The Five Assumptions of Multiple Linear Regression.* Retrieved from Statology: https://www.statology.org/multiple-linear-regression-assumptions/