**D208 Task 1: Linear Regression Modeling**

**Joseph Duszynski**

**ID #011237236**

**A.1. Research Question**

Using the medical data set (WGU, 2024 [1]), this project seeks to answer the question “Which patient variables influence their average daily charge (‘TotalCharge’) and can this data create reliable predictions through a linear regression model?”

**A.2. Goals**

This project aims to find the most influential variables affecting a patient’s average daily charge and build a linear regression model to accurately predict it. These insights can assist hospital administrators and analysts understand what factors drive a patient’s daily average expenses.

**B.1. Summary of assumptions**

Multiple linear regression models are assumed to (Sewell, 2024 [1]):

1. Have independent variables with low correlations between one another (no strong multicollinearity)
2. The independent variables have a linear relationship with the dependent variable. That is, where the first column of the matrix X is 1s to allow for a constant term and the **β** vector contains the parameters of the relationship.
3. The residuals follow a normal distribution with mean 0.
4. An increase in the independent variables improves the model’s explanatory power.

**B.2. Tool benefits**

I chose Python 3.9 for its ease of use, widespread adoption, personal familiarity, speed, multitude of mathematical and statistical packages, and extensive documentation and examples online for debugging and understanding. I prefer the consistent syntax of Python and its packages compared to libraries in R. Additionally, I wrote data cleaning functions in Python for the D206 course, making it convenient to repurpose that work here.

Numpy and pandas were used for handling numerical computations on arrays and dataframes, respectively. They’re effectively mandatory for any of the project to work. I needed matplotlib and seaborn to produce histograms and scatterplots. Scipy.stats was required for calculating z-scores to be used in creating z-score histograms. Statsmodels was used for the linear regression modeling and variance inflation factor calculations. From scikit-learn, various packages were used in the recursive feature elimination procedure.

**B.3. Appropriate technique**

The research question aims to predict numerical values of a continuous dependent variable (‘TotalCharge’) from several independent variables (to be discussed in future sections). The independent variables are all numerical data (categorical variables were one-hot encoded). Multiple linear regression requires numerical independent variables and a numerical target variable with the variable selection and respective estimated parameters chosen to conform to the assumptions of the model from section B.1. As will be later demonstrated, there is a linear relationship between the independent variables and target variable, making multiple linear regression a suitable approach for building a predictive model.

**C.1. Data cleaning**

For accurate modeling, the data should be free of nulls, outliers, and duplicates. To do so, ‘CaseOrder’ and ‘Zip’ are re-expressed as categorical (string) data, .isna.sum() > 0 locates any columns with null values, and .duplicated(keep=False) finds any duplicates. For outliers and column constraints, the functions inspect\_data and outlier\_search were used. .describe() and .value\_counts() give a summary of the data columns. outlier\_search produces a histogram and prints outliers that are outside of the interquartile range or have a z-score with an absolute value above 3.0. See the attached file “linear.py” as well as the code segment below:

# Creates a new column 'Zip\_int64' to back up the old 'Zip' values while adjusting current 'Zip' values to strings with  
# five digits.  
def zip\_to\_str(zip\_col='Zip', df=df\_med):  
 df['Zip\_int64'] = df[zip\_col]  
 df[zip\_col] = df[zip\_col].astype('str')  
 for i in range(5):  
 df[zip\_col].mask(df[zip\_col].str.len() == i, '0' \* (5 - i) + df[zip\_col], inplace=True)  
 print(  
 f"Verifying number of entries in 'Zip' with number of digits other than 5: {len(df.loc[df['Zip'].str.len() != 5, 'Zip'])}\n")  
 print(  
 f"Verifying number of entries in 'Zip' with number of digits exactly 5: {len(df.loc[df['Zip'].str.len() == 5, 'Zip'])}\n")

# Changing 'CaseOrder' and 'Zip' to strings, also verifying there are no duplicates or nulls  
df\_med['CaseOrder'] = df\_med['CaseOrder'].astype('str')  
zip\_to\_str()  
print(f"Checking for columns with null values: {list(df\_med.columns[df\_med.isna().sum() > 0])}\n")  
print("Verifying there are no duplicate entries ('False' indicates not a duplicate):")  
print(df\_med.duplicated(keep=False).value\_counts())  
print("\n")  
  
  
# Dataframe description and value counts  
def inspect\_data(columns, df=df\_med):  
 for col in columns:  
 if (df[col].dtype == 'int64') or (df[col].dtype == 'float64'):  
 print(f"\nNumber of unique values: {len(df[col].unique())}")  
 print(df[col].describe())  
 else:  
 print(df[col].describe())  
 print(df[col].value\_counts())  
 print("\n")  
  
  
# Searches for outliers by IQR and z-scores (defaults to |z| > 3.0) with optional z-score histogram plot  
def outlier\_search(columns, plots=True, z\_bound=3.0, df=df\_med):  
 df\_outliers\_dict = {}  
 df\_zscore\_outl\_dict = {}  
 for column in columns:  
 col\_stats = df[column].describe()  
 q25 = col\_stats['25%']  
 q75 = col\_stats['75%']  
 lower\_bound = q25 - 1.5 \* (q75 - q25)  
 upper\_bound = q75 + 1.5 \* (q75 - q25)  
 df\_outliers = df[(df[column] < lower\_bound) | (df[column] > upper\_bound)]  
 col\_zscore = column + '\_zscore'  
 with pd.option\_context("mode.chained\_assignment", None):  
 df\_outliers[col\_zscore] = stats.zscore(df[column])  
 df\_zscore = pd.DataFrame(stats.zscore(df[column]))  
 df\_zscore\_outl = df\_zscore[abs(df\_zscore[column]) > z\_bound]  
 print("----------------------")  
 print(f"{column}:")  
 print(col\_stats)  
 print("\nZ-scores:")  
 print(df\_zscore.describe())  
 print(f"\nIQR test for outliers has a lower bound of {round(lower\_bound, 3)} and an upper bound"  
 f" of {round(upper\_bound, 3)}")  
 print(f"Z-scores have a lower bound of {-1 \* z\_bound} and an upper bound of {z\_bound}\n")  
 if df\_outliers.empty and df\_zscore.empty:  
 print(f"There are no outliers in the column {column}.")

else:  
 print(f"By IQR, there are {len(df\_outliers)} outliers.")  
 print(df\_outliers[[column, col\_zscore]])  
 print(f"\nBy z-score, there are {len(df\_zscore\_outl)} outliers.")  
 print(df\_zscore\_outl)  
 df\_outliers\_dict[column] = df\_outliers  
 df\_zscore\_outl\_dict[column] = df\_zscore\_outl  
 if plots:  
 plt.hist(df\_zscore)  
 plt.xlabel(column + ' z-score')  
 plt.ylabel('Frequency')  
 plt.show()  
 print("----------------------\n")  
 return df\_outliers\_dict, df\_zscore\_outl\_dict  
  
  
inspect\_data(df\_med.columns)  
outlier\_search(small\_int\_columns)  
outlier\_search(continuous\_columns)  
outlier\_search(item1\_to\_8\_columns)

A screenshot of a computer

Description automatically generated

**C.2. Summary statistics**

Summary statistics output of .describe() for all independent variables and the dependent variable ‘TotalCharge’. Categorical variables additionally include .value\_counts()).

relevant\_columns = ['TotalCharge', 'Initial\_days', 'Children', 'Additional\_charges', 'vitD\_supp', 'Age', 'Population', 'Item1', 'ReAdmis', 'Complication\_risk', 'Initial\_admin', 'BackPain', 'Arthritis', 'Anxiety', 'Reflux\_esophagitis', 'HighBlood', 'Allergic\_rhinitis', 'Hyperlipidemia', 'Diabetes']  
inspect\_data(df\_med[relevant\_columns])

Number of unique values: 9997

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

Number of unique values: 9997

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

Number of unique values: 11

count 10000.000000

mean 2.097200

std 2.163659

min 0.000000

25% 0.000000

50% 1.000000

75% 3.000000

max 10.000000

Name: Children, dtype: float64

Number of unique values: 9418

count 10000.000000

mean 12934.528587

std 6542.601544

min 3125.703000

25% 7986.487755

50% 11573.977735

75% 15626.490000

max 30566.070000

Name: Additional\_charges, dtype: float64

Number of unique values: 6

count 10000.000000

mean 0.398900

std 0.628505

min 0.000000

25% 0.000000

50% 0.000000

75% 1.000000

max 5.000000

Name: vitD\_supp, dtype: float64

Number of unique values: 72

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

Number of unique values: 5951

count 10000.000000

mean 9965.253800

std 14824.758614

min 0.000000

25% 694.750000

50% 2769.000000

75% 13945.000000

max 122814.000000

Name: Population, dtype: float64

Number of unique values: 8

count 10000.000000

mean 3.518800

std 1.031966

min 1.000000

25% 3.000000

50% 4.000000

75% 4.000000

max 8.000000

Name: Item1, dtype: float64

count 10000

unique 2

top No

freq 6331

Name: ReAdmis, dtype: object

ReAdmis

No 6331

Yes 3669

Name: count, dtype: int64

count 10000

unique 3

top Medium

freq 4517

Name: Complication\_risk, dtype: object

Complication\_risk

Medium 4517

High 3358

Low 2125

Name: count, dtype: int64

count 10000

unique 3

top Emergency Admission

freq 5060

Name: Initial\_admin, dtype: object

Initial\_admin

Emergency Admission 5060

Elective Admission 2504

Observation Admission 2436

Name: count, dtype: int64

count 10000

unique 2

top No

freq 5886

Name: BackPain, dtype: object

BackPain

No 5886

Yes 4114

Name: count, dtype: int64

count 10000

unique 2

top No

freq 6426

Name: Arthritis, dtype: object

Arthritis

No 6426

Yes 3574

Name: count, dtype: int64

count 10000

unique 2

top No

freq 6785

Name: Anxiety, dtype: object

Anxiety

No 6785

Yes 3215

Name: count, dtype: int64

count 10000

unique 2

top No

freq 5865

Name: Reflux\_esophagitis, dtype: object

Reflux\_esophagitis

No 5865

Yes 4135

Name: count, dtype: int64

count 10000

unique 2

top No

freq 5910

Name: HighBlood, dtype: object

HighBlood

No 5910

Yes 4090

Name: count, dtype: int64

count 10000

unique 2

top No

freq 6059

Name: Allergic\_rhinitis, dtype: object

Allergic\_rhinitis

No 6059

Yes 3941

Name: count, dtype: int64

count 10000

unique 2

top No

freq 6628

Name: Hyperlipidemia, dtype: object

Hyperlipidemia

No 6628

Yes 3372

Name: count, dtype: int64

count 10000

unique 2

top No

freq 7262

Name: Diabetes, dtype: object

Diabetes

No 7262

Yes 2738

Name: count, dtype: int64

The numerical independent variables to be used in the initial model:

* 'Initial\_days'
* 'Children'
* 'Additional\_charges'
* 'vitD\_supp'
* 'Age'
* 'Population'
* 'Item1'

And the categorical independent variables to be used in the initial model:

* 'ReAdmis'
* 'Complication\_risk'
* 'Initial\_admin'
* 'BackPain'
* 'Arthritis'
* 'Anxiety'
* 'Reflux\_esophagitis'
* 'HighBlood'
* 'Allergic\_rhinitis'
* 'Hyperlipidemia'
* 'Diabetes'

These initial variables were selected by looking at the correlation matrix values restricted to the target variable ‘TotalCharge’ in descending order. See below.

def corr\_search(column, thresh=-1.0, df=df\_med):  
 sorted\_corr = df.select\_dtypes(exclude=['object']).corr()[column].sort\_values(ascending=False, key=lambda x: abs(x))  
 print(sorted\_corr[abs(sorted\_corr) >= thresh])  
 print("\n")  
  
  
corr\_search(model\_dep\_var)  
corr\_search('Initial\_days', 0.03)  
corr\_search('ReAdmis\_Yes', 0.03)  
corr\_search('Initial\_admin\_Emergency Admission', 0.03)  
  
print(df\_med.select\_dtypes(exclude=['object']).corr()[model\_dep\_var].sort\_values(ascending=False,  
 key=lambda x: abs(x)).index[1:19])

With output (note that categorical variables have been one-hot encoded and renamed accordingly):

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Selecting the highest 18 independent variables based on their correlation to ‘TotalCharge’ produces the initial model’s variables mentioned above. ‘Complication\_risk\_Low’ was included for completeness (as ‘Complication\_risk’ had values of ‘Low’, ‘Medium’, and ‘High’, with ‘High’ being dropped by the encoder). Although ‘Diabetes\_Yes’ was at first removed, later optimization attempts found it made a drastic improvement in the model so it’s included in the initial model as well. This will be discussed in greater detail in future sections.

Notably, as depicted in the plots shown in the following section, ‘TotalCharge’ has a bimodal distribution with the left peak from ~2000 to ~5000 and the second peak from ~5000 to ~9200. Its mean and median are both at ~5200, which as one of the lowest frequencies across the entire distribution’s range. ‘Initial\_days’ is also a bimodal distribution separated at ~30 days.

‘Initial\_admin’ is ~50% emergency admissions, ~25% elective, and ~25% observation. ‘Complication\_risk’ is ~45% medium, ~34% high, and ~21% low. For the binary ‘yes’/’no’ variables such as ‘Diabetes’ and ‘BackPain’, the majority response is generally at ~59-72% (with the minority response its complement of ~28-41%).

**C.3. Visualizations**

A graph of blue and white bars

Description automatically generated with medium confidence

A graph with blue bars

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated

A graph of a number of blue rectangular objects

Description automatically generated with medium confidence

A graph with blue bars

Description automatically generated

A graph of age and age

Description automatically generated

A graph of population

Description automatically generated

A graph with blue bars

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph with blue bars

Description automatically generated

A graph of a bar chart

Description automatically generated with medium confidence

A graph of a bar graph

Description automatically generated with medium confidence

A graph with blue rectangles

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph of a patient with blue bars

Description automatically generated with medium confidence

A graph of a scatter plot

Description automatically generated

A graph of blue dots

Description automatically generated

A diagram of a scatter plot

Description automatically generated

A graph of a scatter chart

Description automatically generated

A diagram of a scatter plot

Description automatically generated

A diagram of a scatter plot

Description automatically generated

A graph of blue dots

Description automatically generated

A graph with numbers and lines

Description automatically generated with medium confidence

A graph of scatter plot of total charge versus complication

Description automatically generated

A graph of a scatter chart

Description automatically generated

A graph of a scatter chart

Description automatically generated

A graph of a scatter plot

Description automatically generated

A graph with blue circles

Description automatically generated

A graph of a scatter plot

Description automatically generated

A graph of a scatter plot

Description automatically generated

A graph with blue circles

Description automatically generated

A graph of a scatter plot

Description automatically generated

A graph of a scatter chart

Description automatically generated

**C.4. Data transformation**

To appropriately modify and transform the data, the data first must be cleaned as discussed in C.1. No nulls or duplicates were found, and the detected outliers were determined to be genuine (accurate) input with high z-scores. This will be demonstrated in the presentation in part G.

For this linear regression model, the only transformations necessary are one-hot encoding on categorical columns (as linear regression requires numerical inputs) and the addition of a ‘model\_constant’ column of 1s to give the model a non-zero y-intercept.

The function one\_hot\_encoder performs one-hot encoding on categorical variables and creates new columns (e.g. ‘Diabetes’ becomes ‘Diabetes\_Yes’ with a ‘No’ value for ‘Diabetes’ corresponding to 0 in the ‘Diabetes\_Yes’ column). Refinement of the model when searching for multicollinearity required the manual creation of ‘Complication\_risk\_High’ (the one-hot encoding dropped the ‘High’ value and only used ‘Complication\_risk\_Medium’ and ‘Complication\_risk\_Low’), which will be discussed in sections D. and E. Creating a column of 1s was done simply through df\_med['model\_constant'] = 1. See the attached file “linear.py” and the code segment below:

global\_encoded\_columns = []  
# one hot encoding that maintains a list of encoded columns in global\_encoded\_columns  
def one\_hot\_encoder(columns, df=df\_med):  
 for column in columns:  
 df\_one\_hot\_col = pd.get\_dummies(df[column], drop\_first=True).astype('int32')  
 for col in df\_one\_hot\_col.columns:  
 col\_name = f'{column}\_' + col  
 if col\_name not in global\_encoded\_columns:  
 global\_encoded\_columns.append(col\_name)  
 df[col\_name] = df\_one\_hot\_col[col]  
  
  
# 'Marital', 'TimeZone', and 'State' were found to have very low coefficients, but are not depicted due to the  
# large number of additional columns they created. The remaining categorical variables are to be one hot encoded.  
columns\_to\_encode = ['Area', 'Gender', 'Initial\_admin', 'Complication\_risk', 'Services',  
 'ReAdmis', 'Soft\_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes',  
 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Asthma']  
one\_hot\_encoder(columns\_to\_encode)  
print("Verifying appended columns in dataframe:")  
print(df\_med.columns)  
print(f"\nColumns encoded: {global\_encoded\_columns}\n")

# Manually creating 'Complication\_risk\_High' column as the Medium and Low values were correlated and produced inferior  
# model results. Creating a vector of 1s for a constant term.  
df\_med['Complication\_risk\_High'] = 1 - df\_med['Complication\_risk\_Medium'] - df\_med['Complication\_risk\_Low']  
df\_med['model\_constant'] = 1

**C.5. Prepared data set**

See attached file “medical\_transformed.csv”.

**D.1. Initial model**

Initial model created from:

# Initial model independent variables  
model\_indp\_var = ['model\_constant', 'Initial\_days', 'ReAdmis\_Yes', 'Initial\_admin\_Emergency Admission',  
 'Complication\_risk\_Medium', 'Complication\_risk\_Low', 'Initial\_admin\_Observation Admission',  
 'BackPain\_Yes', 'Arthritis\_Yes', 'Anxiety\_Yes', 'Additional\_charges', 'Reflux\_esophagitis\_Yes',  
 'Children', 'HighBlood\_Yes', 'Item1', 'Population', 'Allergic\_rhinitis\_Yes', 'Hyperlipidemia\_Yes',  
 'Diabetes\_Yes', 'vitD\_supp', 'Age']  
model\_dep\_var = 'TotalCharge'  
X\_0 = df\_med[model\_indp\_var]  
y\_0 = df\_med[model\_dep\_var]  
  
  
# Linear regression model for target y ('TotalCharge' in this case) with independent variables X (see model\_indp\_var)  
def lin\_model(y, X):  
 model = sm.OLS(y, X)  
 results = model.fit()  
 print(results.summary())  
 residuals = results.resid  
 print(f"\nResidual Standard Error: {np.sqrt(results.mse\_resid)}\n")  
  
 sns.heatmap(X.corr(), annot=True)  
 plt.show()  
  
 y\_pred = results.predict(X)  
 sns.scatterplot(x=y\_pred, y=residuals)  
 plt.xlabel('Predicted value of MonthlyCharge')  
 plt.ylabel('Residuals')  
 plt.title('Scatter plot of residuals vs predicted values')  
 plt.show()  
  
 sm.qqplot(residuals)  
 plt.title('QQ plot of residuals')  
 plt.show()  
  
 vif\_data = pd.DataFrame({'feature': X.columns, 'VIF': [variance\_inflation\_factor(X.values, i) for i in range(len(X.columns))]})  
 print(vif\_data)

Output:

A screenshot of a computer screen

Description automatically generated

For reference, without the inclusion of ‘Diabetes\_Yes’, this is the resulting model:

A screenshot of a computer screen

Description automatically generated

Without the inclusion of ‘Diabetes’, the initial model’s residual standard error is significantly higher (33.5 compared to 2.62\*10-4) and its AIC increases dramatically from -1.365\*105 to 9.865\*104. Despite its low correlation value from the earlier results to determine the initial set of independent variables, it’s an important component to retain.

**D.2. Justification of model reduction**

The initial model has a very high R2 value, but given the number of independent variables and strong correlation between ‘Initial\_days’ and ‘TotalCharge’ (see its scatter plot in section C.3. and the 0.988 correlation value found in section C.2.), this is not the best metric to assess model quality. The condition number is extraordinarily high (over 105), indicating extensive multicollinearity. Additionally, many of the coefficients are vanishingly small with large p-values.

First, inspecting a heatmap of the correlations.

A graph of a graph

Description automatically generated

The correlation value output in section C.2. captures the most relevant relationships. ‘Additional\_charges’ has high correlation values with ‘Age’ and ‘HighBlood\_Yes’. Notably, the following relationship can be found in its scatter plot.

A graph of a number of dots

Description automatically generated with medium confidence

While a secondary linear regression model could be done to absorb the ‘Additional\_charges’ variable into ‘Age’ and ‘HighBlood\_Yes’ (and ‘Age \* High\_Blood\_Yes’), its coefficient in the model is ~10-9. It takes values from ~3000 to ~30000, but since ‘TotalCharge’ ranges from ~2000 to ~9200, its contribution is infinitesimal.

The target variable has strong correlations to both ‘Initial\_days’ and ‘ReAdmis\_Yes’, but ‘Initial\_days’ and ‘ReAdmis\_Yes’ have a strong correlation to each other, so a significant amount of the multicollinearity results from including both of these variables. ‘ReAdmis\_Yes’ has a very small parameter and has a negligible contribution.

There’s a strong negative correlation between ‘Initial\_admin\_Emergency Admission’ and ‘Initial\_admin\_Observation Admission’, but it’s worth noting that the p-value is extremely high and the coefficient is vanishingly small on ‘Initial\_admin\_Observation Admission’.

A strong negative correlation also exists between ‘Complication\_risk\_Medium’ and ‘Complication\_risk\_Low’. Since their coefficients are virtually identical (-413.49), it indicates this could be simplified by manually introducing a one-hot encoding for ‘Complication\_risk\_High’.

df\_med['Complication\_risk\_High'] = 1 - df\_med['Complication\_risk\_Medium'] - df\_med['Complication\_risk\_Low']

This exchange would shift the constant term down by ~413.49, invert the sign of the coefficient, and reduce an unnecessary predictor variable.

Reviewing other variables with high p-values and small coefficients, ‘Age’, ‘vitD\_supp’, ‘Population’, ‘Item1’, and ‘Children’ all appear to be safe to exclude.

To justify the removal of these variables, we’ll look at variance inflation factors as well as a recursive feature elimination procedure. High VIF (above ~5-10) indicates high multicollinearity.

A screenshot of a computer program

Description automatically generated

‘Additional\_charges’ and ‘Age’ should be removed given the above, while the high VIF on blood pressure is likely due to the multicollinearity between ‘Additional\_charges’, ‘Age’, and ‘HighBlood\_Yes’.

Recursive feature elimination (RFE) recursively eliminates features from the model using a regression function, removing the least important feature at each step.

Performing RFE (set to keep 13 features):

Index(['Initial\_days', 'Initial\_admin\_Emergency Admission',

'Complication\_risk\_Medium', 'Complication\_risk\_Low', 'BackPain\_Yes',

'Arthritis\_Yes', 'Anxiety\_Yes', 'Additional\_charges',

'Reflux\_esophagitis\_Yes', 'HighBlood\_Yes', 'Allergic\_rhinitis\_Yes',

'Hyperlipidemia\_Yes', 'Diabetes\_Yes'],

dtype='object')

{'model\_constant': 9, 'Initial\_days': 1, 'ReAdmis\_Yes': 5, 'Initial\_admin\_Emergency Admission': 1, 'Complication\_risk\_Medium': 1, 'Complication\_risk\_Low': 1, 'Initial\_admin\_Observation Admission': 8, 'BackPain\_Yes': 1, 'Arthritis\_Yes': 1, 'Anxiety\_Yes': 1, 'Additional\_charges': 1, 'Reflux\_esophagitis\_Yes': 1, 'Children': 4, 'HighBlood\_Yes': 1, 'Item1': 3, 'Population': 6, 'Allergic\_rhinitis\_Yes': 1, 'Hyperlipidemia\_Yes': 1, 'Diabetes\_Yes': 1, 'vitD\_supp': 7, 'Age': 2}

Ranking: [9 1 5 1 1 1 8 1 1 1 1 1 4 1 3 6 1 1 1 7 2]

R^2; 0.999999810120033

MSE: 0.9043382845751022

Recursive feature elimination suggests removing (with higher numbers being removed first):

* 'model\_constant': 9
* 'ReAdmis\_Yes': 5
* 'Initial\_admin\_Observation Admission': 8
* 'Children': 4
* 'Item1': 3
* 'Population': 6
* 'vitD\_supp': 7
* 'Age': 2

To review, prior to inspecting variance inflation and running recursive feature elimination, the variables with negligible contributions were identified as:

* ‘Additional\_charges’
* ‘ReAdmis\_Yes’
* ‘Initial\_admin\_Observation Admission’
* ‘Age’
* ‘vitD\_supp’
* ‘Population’
* ‘Item1’
* ‘Children’

This list agrees almost exactly with the results of RFE, with the exception of suggesting the constant be removed in place of ‘Additional\_charges’. Given the high variance inflation on ‘Additional\_charges’, it should be removed. The constant is better off retained due to an improved distribution of the residuals and lower AIC, which will be discussed further in section E.1.

When merging ‘Complication\_risk\_Low’ and ‘\_Medium’ into ‘\_High’ and removing the variables discussed above, we’re left with the following 11 variables and a constant term:

* 'model\_constant'
* 'Initial\_days'
* 'Initial\_admin\_Emergency Admission'
* 'Complication\_risk\_High'
* 'BackPain\_Yes'
* 'Arthritis\_Yes'
* 'Anxiety\_Yes'
* 'Reflux\_esophagitis\_Yes'
* 'HighBlood\_Yes'
* 'Allergic\_rhinitis\_Yes'
* 'Hyperlipidemia\_Yes'
* 'Diabetes\_Yes'

Removing additional variables produces higher residual standard errors and less desirable residual plots, which will be discussed in section E.1.

**D.3. Reduced linear regression model**

The resulting reduced model:

A screenshot of a computer

Description automatically generated

**E.1. Model comparison**

The target ‘TotalCharge’ has such a strong correlation to ‘Initial\_days’ that removing inconsequential variables with small coefficients would have very small effects on the remaining variables. Inspecting each coefficient, standard error, and model statistic, the refined model summary is virtually identical to the summary of the initial model (when restricting to the variables that were retained). The primary improvement comes from a reduction in the condition number (initially ~105 reduced to 151) and removal of most of the multicollinearity. Note that the change in the constant resulted from merging the ‘Complication\_risk’ variables into ‘Complication\_risk\_High’. The RSE is 2.62\*10-4 and the AIC is -1.37\*105 for both models, indicating there wasn’t much change. Given the small coefficients on the removed variables, this is to be expected. The model has been refined with a smaller condition number, low multicollinearity, and simpler to understand.

While it’s tempting to reduce the model further, removal of any other variable increases the residual standard error, raises the AIC, and creates residual plots less favorable that clearly demonstrate they shouldn’t have been removed.

For instance, running RFE on the refined model suggests removing ‘Reflux\_esophagitis\_Yes’, ‘Allergic\_rhinitis\_Yes’, and ‘Arthritis\_Yes’, which results in:

A screenshot of a computer

Description automatically generated

A graph of blue dots

Description automatically generated with medium confidence

Contrast the above residuals plot (and their magnitudes) with the one shown in section E.2. The six horizontal lines introduced indicate three clusters that neatly align with the three removed binary categorical variables. Further, the condition number is virtually unchanged at the expense of a higher RSE of 54.44 and AIC increasing to ~1.08\*105.

When the constant is removed:

A diagram of a scatter plot

Description automatically generated

A screenshot of a computer

Description automatically generated

The RSE is now at ~626 with an AIC of ~1.6\*105.

For the refined model of 11 variables, it’s strongly favored for its extraordinarily low residuals, an RSE that is virtually 0, very low AIC, F-statistic p-value of 0.00, and coefficients with infinitesimal standard errors.

**E.2. Output and calculations**

Residual standard error of refined model:

Residual Standard Error: 0.00026189898042647004

Residual plot of refined model:

A diagram of a scatter plot

Description automatically generated

QQ plot:

A blue line graph with white text

Description automatically generated

**E.3. Code**

See attached file “linear.py”.

**F.1. Results**

The model produces the following multilinear equation:

*‘TotalCharge’ = 81.9378 \* 'Initial\_days' + 512.3232 \* 'Initial\_admin\_Emergency Admission' + 413.4943 \* 'Complication\_risk\_High ' + 85.1459 \* 'BackPain\_Yes ' + 71.9509 \* 'Arthritis\_Yes' + 86.1143 \* 'Anxiety\_Yes' + 59.6802 \* 'Reflux\_esophagitis\_Yes' + 112.3232 \* 'HighBlood\_Yes' + 60.5811 \* 'Allergic\_rhinitis\_Yes' + 93.9901 \* 'Hyperlipidemia\_Yes' + 75.2032 \* 'Diabetes\_Yes' + 1855.6859*

In words, it indicates a patient’s total charge will change by:

* 81.9378 for every additional day in ‘Initial\_days’
* 512.3232 if the patient was an emergency admission (otherwise 0)
* 413.4943 if the patient has a high complication risk (otherwise 0)
* 85.1459 for patients with back pain
* 71.9509 for patients with arthritis
* 86.1143 for patients with anxiety
* 59.6802 for patients with reflux esophagitis
* 112.3232 for patients with high blood pressure
* 60.5811 for patients with allergic rhinitis
* 93.9901 for patients with hyperlipidemia
* 75.2032 for patients with diabetes
* Constant term of 1855.6859 is a minimum achieved by patients who stayed 0 days initially and have none of the conditions above

This model is statistically significant as it has an R2 of 1.00, F-statistic p-value of 0.00, very low AIC of ~1.4\*105, RSE of 2.6\*10-4, omnibus and Jarque-Bera probabilities well below a threshold of 0.05, p-values of 0.000 for all coefficients, coefficient standard errors on the order of 10-5, and a comparatively low condition number of 151. The residuals are virtually 0 for predicted values below ~5000 with random noise introduced in the second peak (above ~5000), but even then they’re under ~10-3 for a target variable ranging from ~2000 to ~9200.

Practically, this model is an almost perfect prediction for a patient’s daily average billing from the eleven variables used in the refined model. Ten of which are simple binary determinations that can be found in a patient’s chart and the numerical one a record of the length of their initial stay. Hospital administrators and analysts can use this model to quickly compute the expected billable expenses for a patient (averaged per day) or even identify potential billing mistakes for patients that significantly deviate from this model (given how small the residuals are).

This model is only as good as its input data. While highly accurate from the existing data, there’s no guarantee this relationship will hold indefinitely, especially since patient and hospital expenses are very complex. The hospital may need to change suppliers, prices may go up (or down), the hospital’s nightly fee per room may change, and patients may start to refuse certain tests and expensive procedures. As it currently stands, this model essentially implies that any patient with one of the ten binary conditions, high blood pressure for instance, has their daily expenses increased by a flat amount. Not all patients will necessarily consent to all tests and procedures, and as there are varying degrees of problems resulting from high blood pressure, it’s difficult to see all patients with high blood pressure having the exact same treatments and exact same additional expenses each day. Data input errors would contaminate the model with unexpected outliers and create billing mistakes for the hospital and patients.

It's strange that the average daily expenses for a patient are linear with the number of initial days stayed in the hospital. Given the way it’s phrased in the data dictionary, it implies a patient’s final bill after leaving the hospital has a quadratic relationship with the number of days spent, which is difficult to comprehend how that would function (with conventional billing practices) and have such small residuals. It also implies that a patient who’s readmitted has expenses that can be chiefly determined by the length of their initial stay, independent of their existing health concern, which also seems unlikely in practice. That said, this model relies on correlations, and despite its high accuracy isn’t causative in either direction of a patient’s bill or their health conditions.

The refined model’s condition number indicates it’s still sensitive to initial conditions, so should errors later be found or future data added to the model that doesn’t conform to the existing data set, the model’s parameters could change to a nontrivial degree.

The model could have been improved with more granular detail on patient health conditions. Rather than entering anxiety, obesity, and other conditions in a binary way, they could have been done with an ordinal scale or numerical data in the case of weight, height, and blood pressure.

**F.2. Recommendations**

In seeking an answer to the research question what variables influence a patient’s ‘TotalCharge’ and how they can be used to predict it, a multiple linear regression model was found to have incredibly high accuracy and statistical significance, relying on the input variables discussed in the previous section F.1.

Since the residuals are a vanishingly small percentage of a patient’s average daily billable expenses, hospital administrators and analysts can use the model for extraordinarily accurate short-term revenue forecasts. Additionally, breakdowns in the model can point to potential changes in hospital billing practices and prices that would require the model to be updated. Patients who have bills significantly different from the model’s prediction could flag possible billing mistakes or, if there is no mistake, serve as valuable information for a necessary adjustment to the model.

The model could be made more robust for future data that will inevitably diminish its accuracy by increasing the data set’s columns and level of detail. For instance, numerical blood pressure readings and hospital prices for specific procedures would be very helpful in producing better approximations that are relatively future proof (i.e. a lower condition number).

**G. Panopto Demonstration**

See the attached link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b42634d1-caed-4c96-9492-b15a00206403>

Addendum discussing model coefficient interpretation: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=cad7478e-2824-4c41-a416-b15a0025bb8a

**H. Sources of third-party code**

**1.** WGU. 2024. D208 Predictive Modeling “Medical Data Dictionary and Data Set”. Medical Data and Dictionary Files. Retrieved April 17, 2024, from <https://access.wgu.edu/ASP3/aap/content/g9rke9s0rlc9ejd92md0.html>.

**2.** Boeye, Jeroen. 2024. DataCamp "Dimensionality Reduction in Python". Chapter 3. Retrieved April 21, 2024, from https://campus.datacamp.com/courses/dimensionality-reduction-in-python/feature-selection-ii-selecting-for-model-accuracy?ex=1.

**I. Sources**

**1.** Sewell, William. 2024. WGU D208 Predictive Modeling Webinar Episode 25. Retrieved April 20, 2024, from https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/\_layouts/15/Doc.aspx?sourcedoc=%7BE500E42D-9568-4430-BF11-A23D245FDFD5%7D.