Predicting Total Daily Charges for Hospital Patients with Multiple Regression

Brandon Scholer

Department of Information Technology, Western Governors University

D208: Predictive Modeling

Dr. Straw

November 10, 2021

With a data set including patient information such as medical conditions, medical information, and demographic information, can the total average cost that the patient will be charged daily be predicted? The hope is that any statistically significant variables, or relationships between the variables, can be isolated with the analysis, while the model itself can be evaluated for its prediction capabilities.

Since the response variable in this analysis is a continuous one, with various types of predictors, the logical choice is to use multiple regression utilizing ordinary least squares (OLS). To perform OLS, linearity between the independent and dependent variables needs to exist, and there must also be normality in the distribution of the variables. Multicollinearity is an issue, so the independent variables should not be highly correlated. The variance of the error terms should be similar across the variables to fulfill the homoscedasticity requirement of OLS as well. Python was the language of choice for this analysis because it has "a great number of data-oriented feature packages that can speed up and simplify data processing, making it time-saving" (Zhidkov, 2020). Python allowed for the adjustment of packages at different points in the analysis to better support specific goals at the moment.

Variables that didn't add any specific information to the data were scrubbed to prepare it properly. For example, variables like CaseOrder, Customer_id, Interaction, and UID were all fields used as identifiers for the hospital but added nothing to the data set. City, County, Zip, State, TimeZone, Lat, Lng, and Job were all variables that overlapped with each other or required further analysis by themselves, placing them outside the scope of this particular analysis. A number following the word Item was the name for several variables, and this was adjusted so that the variable names better represented the variables they were describing. Strings of 'Yes' or 'No' were then converted to Boolean values for more accessible data processing.

Certain variables also contained strings that represented a category such as 'Widowed' or 'Never Married' in the Marital variable, so these variables were converted to the type 'Category' for easier handling by Python. Figure 1 contains the code that was used to prepare these data steps.

The summary statistics have been split up by data type as they are held in the data frame, as seen in Figure 2. Five is the maximum number of categories seen in the categorical variable section, with three being the minimum, which is seen across most of the categorical variables. Five categories are only seen in a single variable: Marital. Three of the variables (Initial_admin, Gender, and Services) have a single variable that contains over half of their observations. Population seems right-skewed, with a mean towards the lower end at 10,000, while its range extends upward of 122,000. Full_meals_eaten and Children also have means that fall towards the low end of their respective ranges, while the rest of the discrete variables have means falling near the center of their range. 'No' was the more likely outcome in every binary variable, except one (Overweight). Most of the means of the continuous variable fell either slightly left or right of center, except Income, in which the mean fell significantly closer to the low end of the range.

The histograms seen in Figure 3 show the univariate analysis for the Population, Area, Children, Age, Income, Marital, Gender, ReAdmis, and VitD_levels. As seen, the only numerical, non-binary variable with an entirely normal distribution is VitD_levels. Age is evenly distributed through the entire plot, but Population, Children, and Income all are right-skewed. Figure 4 provides insight into the Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, Complication_risk, and Overweight variables. Here Doc_visits follow a normal distribution, but Full_meals_eaten and vitD_supp are also right-skewed. The rest are either categorical, with one category outpacing the rest, or binary, with one side taking the most observations. Figure 5 shows variables that are almost entirely binary, with one variable (Services) categorical. Again, the binary variables are mostly lopsided. Two of the categories comprise most of the Services variable's four categories.

The variables in Figure 6 are interesting in that most are normally distributed, except three, one of which is in the dependent variable. The Initial_days and TotalCharge (the dependent variable) have a bimodal distribution, while Additional_charges is right-skewed. This dependent variable does violate the assumption of the normal distribution; however, "by the law of large numbers and the central limit theorem, the ordinary least squares (OLS) estimators in linear regression technique still will be approximately normally distributed around the true parameter values, which implies the estimated parameters and their confidence interval estimates remain robust" (Li, Wong, Lamoureux, & Wong, 2012). Meaning a data set with a sufficiently large sample size should remain valid, even though it may violate the normal distribution assumption. Figure 7 contains the last two univariate histograms, both of which are normally distributed.

Figure 8 shows the bivariate relationship between the dependent variable, TotalCharge, and the independent variables Population, Area, Children, Age, Income, Marital, Gender, ReAdmis, and VitD_levels in a histogram. There is no clear evidence of a linear relationship between the continuous independent and dependent variables in these histograms. There may be some linearity in the dense parts of these plots, but it is not apparent that this is the case. ReAdmis is an exciting plot in that it shows some clear separation between the lower and higher end of the dependent variable compared to the true or false value of the independent variable. In Figure 9, there are also no clear linear patterns between the independent variables Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, Complication_risk, and Overweight. The variables in Figure 10 are either binary or categorical, which doesn't give insight into linearity but shows the clusters of observations around 7500 and 3000 for all the independent variables compared to TotalCost. Seven of the nine histograms in Figure 11 don't offer a clear linear relationship between the independent variables and TotalCharge; however, Initial_days has an apparent linear relationship with TotalCost. Figure 12 contains the last two histograms, and again there is no clear linear relationship between independent and dependent variables. To ensure there is no multicollinearity in the model, variance inflation factor (VIF) was used to identify variables that could potentially be collinear. The code seen in Figure 13 establishes a VIF for each variable, then removes variables with a VIF greater than 5, which resulted in the variables Age, HighBlood, and Additional_charges all being removed. These all also had very few discernable linearity patterns in their bivariate analysis.

The first model had 43 degrees of freedom with dummy variables included, as seen in Figure 14. Thirty-one of these variables were not statistically significant at the 0.05 significance level (alpha). The R-squared and Adjusted R-squared levels were both 0.999. an almost perfect Indicating regression model, however with levels that high, it more likely means that there is a good chance of overfitting occurring—also, the Cond. No. in the bottom is high (as shown in a warning at the bottom), which indicated either multicollinearity or an issue with the numbers, likely caused by some non-linear trends seen in the bivariate analysis. Figure 15 shows the root-mean-square error (RMSE) code which gives this model an RMSE of 55.63. This value is then multiplied by the mean of the observed values in y_test, which offers 0.01 or an error rate of roughly 1%. The residuals can be seen in Figure 16. While the residuals are distributed evenly among the x-axis above and below the regression line, they tend to cluster in two areas along the y-axis, giving an odd pattern, though still resulting in a zero-sum.

It was run through a backward elimination technique that pulled the variable with the highest p-value and removed it from the model to reduce this model. The Adjusted R-squared from the new model was then compared with the previous model's Adjusted R-squared. The removed variable was either left out or reintroduced depending on the highest Adjusted R-squared, as seen in Figure 17. This process continued until the model with the highest Adjust R-squared was produced. The reduced model, as seen in Figure 18, brought the degrees of freedom down to 25. At the same time, the R-squared and Adjusted R-squared remained at 0.999, the Cond. No. dropped to 404, and the warning disappeared. Three of the variables no longer seen in the reduced model are Population, Income, and VitD_levels, all of which appeared to be non-linear in the bivariate analysis, likely contributing to such a high Cond. No. in the original model. RMSE of this model is roughly the same as the previous model at 55.17, or 1% when compared to the mean of observations seen in Figure 19. Figure 20 shows that the residuals are largely the same compared to the original model's residuals. In contrast, Figure 21 shows how close the predictions from the reduced model are to the actual values from the y_test variable.

The regression equation for the reduced model is:

$$Y = 2322.76 + 1.12(ReAdmis) + .64(Full\ meals\ eaten) + 2.71(Overweight) + 73.30(Arthritis) + 73.94(Diabetes) + 92.96(Hyperlipidemia) + 86.68(BackPain) + 87.12(Anxiety) + 61.28(Allergic\ Rhinitis) + 60.17(Reflux\ Esophagitis) + 81.92(Initial\ days) - 0.86(Timely\ visits) - 0.86(Reliability) - 1.74(Options) + 1.02(Courteous) + 2.20(Area\ Suburban) + 1.53(Area\ Urban) - 4.77(Marital\ Never\ Married) - 1.60(Marital\ Separated) - 4.31(Marital\ Widowed) + 1.73(Gender\ Nonbinary) + 509.78(Initial\ admin\ Emergency\ Admission) -$$

$416.56(Complication\ risk\ Low) - 413.20(Complication\ risk\ Medium) -$
$0.51(Services\ CT\ Scan)$

Assuming that the currently assessed variable is the only not fixed variable, then the independent variable will increase by the coefficient for each increase in the variable by one. So if ReAdmis increases by one, then TotalCharge will increase by 1.12. If the coefficient is negative, then the dependent variable will decrease by that amount, such that if Marital_Widowed increased by 1, then TotalCharge will decrease by 4.31.

The Adjusted R-Square of this model is phenomenal at 0.999; however, it is so high that the model likely suffers from overfitting. The predictions of the test set fall within roughly 1%, but this may only apply to the data on hand due to said overfitting. The reduced model could not improve upon the original model without becoming a perfect model of the information; thus, the reduced model was capable of reducing the number of variables it used to achieve the same reliability. While the reduced model could accomplish this, too many variables didn't fit either the normality assumption, the linearity assumption, or both. The recommended course would be to either use a different type of analysis that doesn't require linearity and normal distribution or gather more information that follows those assumptions and refit the model to include the new data.

References

Li, X., Wong, W., Lamoureux, E. L., & Wong, T. Y. (2012). Are Linear Regression Techniques Appropriate for Analysis When the Dependent (Outcome) Variable Is Not Normally Distributed? *Investigative Ophthalmology & Visual Science*, 3082-3083.

Zhidkov, R. (2020, January 13). *Why Python is Essential for Data Analysis.* Retrieved from RTInisghts: https://www.rtinsights.com/why-python-is-essential-for-data-analysis/

Figure 1

Data Preparation Phase

```
# Read in the .csv file that contains the data
med_rec = pd.read_csv("C:/Users/clown/OneDrive/Documents/MS Data Analytics/Predictive Modeling/Data/medical_clean.csv")
```

```
# Check for missing values
med_rec.isnull().sum().sum()
```

```
0
```

```
# Drop columns that contain individual indentification that isn't necessary to the model
# and rename vague column names to be more descriptive.
med_rec = med_rec.drop(['CaseOrder','Customer_id','Interaction','UID', 'City','County','Zip','State', 'TimeZone','Lat','Lng','Job'], axis=1)
med_rec = med_rec.rename({'Item1':'Timely_admission','Item2':'Timely_treatment','Item3':'Timely_visits','Item4':'Reliability',
               'Item5':'Options','Item6':'Hours_treatment','Item7':'Courteous','Item8':'Active_listening'}, axis='columns')
```

```
# Convert No/Yes to 0/1 in binary variables
boo = ['ReAdmis','Soft_drink','HighBlood','Stroke','Overweight','Arthritis','Diabetes','Hyperlipidemia',
       'BackPain','Anxiety','Allergic_rhinitis','Reflux_esophagitis','Asthma']
med_rec[boo] = med_rec[boo].replace({'No':False, 'Yes':True})
```

```
# Convert data types to category as appropriate to make analysis easier
cat = ['Area','Marital','Gender','Initial_admin','Complication_risk','Services']
med_rec[cat] = med_rec[cat].astype('category')
```

Note. This figure shows the Python Code in JupyterLab that imports the original CSV file, checks for missing values, remove unnecessary data, and organizes the data so that it is simpler to work with before the prepared data is out into another CSV file.

Figure 2

Summary Statistics for Predictors by Type

In [8]:

```
# Summary statistics for catigorical variables
med_rec.describe(include=['category'])
```

Out[8]:

	Area	Marital	Gender	Initial_admin	Complication_risk	Services
count	10000	10000	10000	10000	10000	10000
unique	3	5	3	3	3	4
top	Rural	Widowed	Female	Emergency Admission	Medium	Blood Work
freq	3369	2045	5018	5060	4517	5265

In [21]:

```
# Summary statistics for discrete and
# ordinal variables of the data type int64
med_rec.describe(include=['int64'])
```

Out[21]:

	Population	Children	Age	Doc_visits	Full_meals_eaten	vitD_supp	Timely_admission	Timely_treatment	Timely_visits	Reliability	Options	Hours_treatment	Courteous	Active_listening
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	9965.253800	2.097200	53.511700	5.012200	1.001400	0.398900	3.518800	3.506700	3.511100	3.515100	3.496900	3.522500	3.494000	3.509700
std	14824.758614	2.163659	20.638538	1.045734	1.008117	0.628505	1.031966	1.034825	1.032755	1.036282	1.030192	1.032376	1.021405	1.042312
min	0.000000	0.000000	18.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	694.750000	0.000000	36.000000	4.000000	0.000000	0.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
50%	2769.000000	1.000000	53.000000	5.000000	1.000000	0.000000	4.000000	3.000000	4.000000	4.000000	3.000000	4.000000	3.000000	3.000000
75%	13945.000000	3.000000	71.000000	6.000000	2.000000	1.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000
max	122814.000000	10.000000	89.000000	9.000000	7.000000	5.000000	8.000000	7.000000	8.000000	7.000000	7.000000	7.000000	7.000000	7.000000

In [10]:

```
# Summary statistics for the continous variables
med_rec.describe(include=['float64'])
```

Out[10]:

	Income	VitD_levels	Initial_days	TotalCharge	Additional_charges
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	40490.495160	17.964262	34.455299	5312.172769	12934.528587
std	28521.153293	2.017231	26.309341	2180.393838	6542.601544
min	154.080000	9.806483	1.001981	1938.312067	3125.703000
25%	19598.775000	16.626439	7.896215	3179.374015	7986.487755
50%	33768.420000	17.951122	35.836244	5213.952000	11573.977735
75%	54296.402500	19.347963	61.161020	7459.699750	15626.490000
max	207249.100000	26.394449	71.981490	9180.728000	30566.070000

In [11]:

```
# Summary statistics for binary variables
# to include the response variable
med_rec.describe(include=['bool'])
```

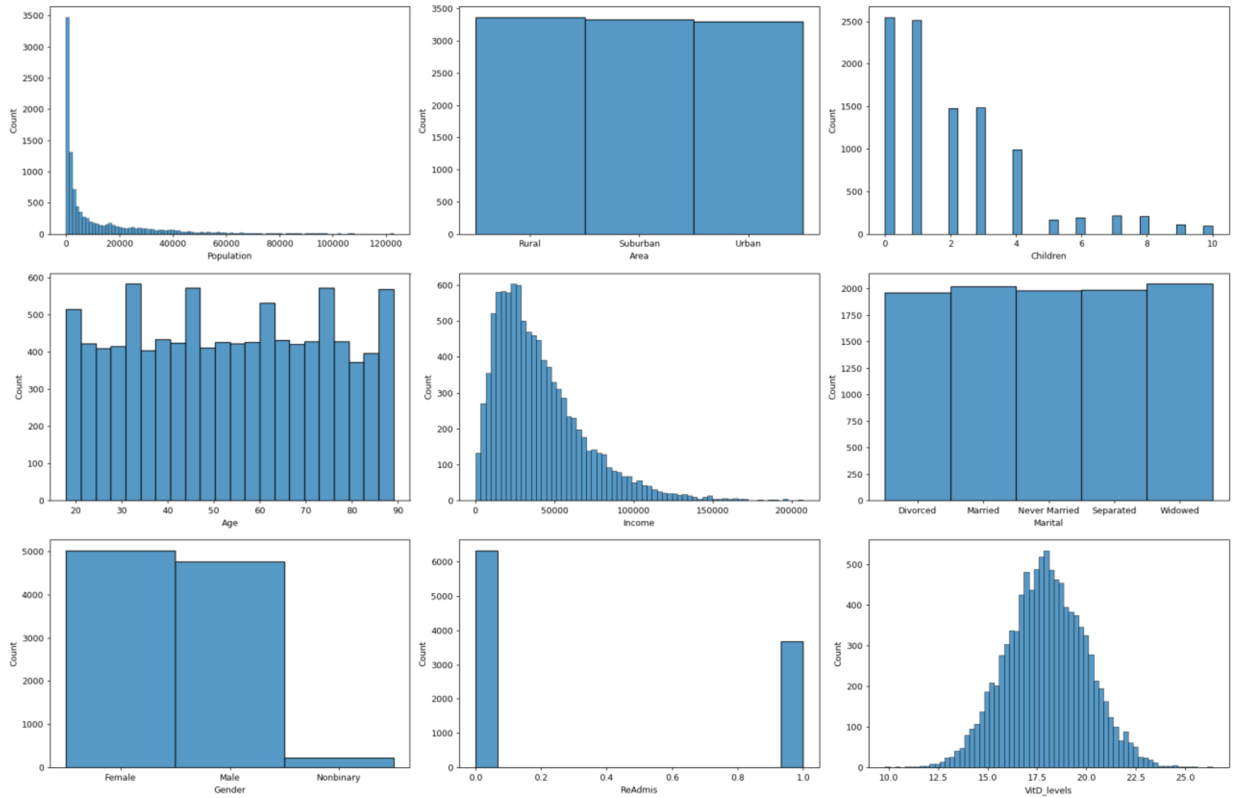
Out[11]:

	ReAdmis	Soft_drink	HighBlood	Stroke	Overweight	Arthritis	Diabetes	Hyperlipidemia	BackPain	Anxiety	Allergic_rhinitis	Reflux_esophagitis	Asthma
count	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
unique	2	2	2	2	2	2	2	2	2	2	2	2	2
top	False	False	False	False	True	False	False	False	False	False	False	False	False
freq	6331	7425	5910	8007	7094	6426	7262	6628	5886	6785	6059	5865	7107

Note. This visual includes the summary statistics for all the variables in the data set (to include the dependent variable TotalCharge), sorted by variable type: categorical, discrete, continuous, and binary.

Figure 3

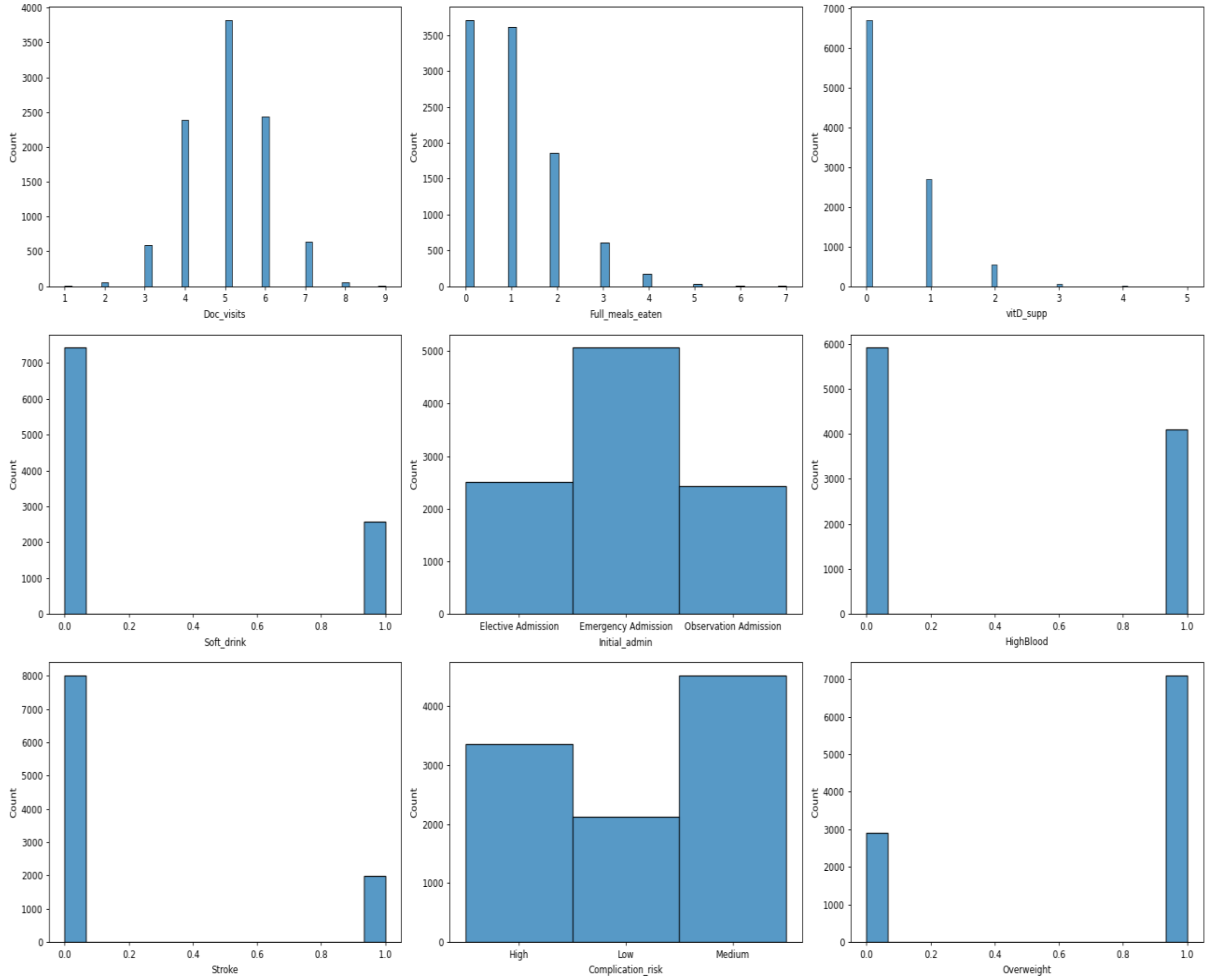
Univariate Visualizations for the First Set of Variables



Note. This visualization shows the univariate bar charts for variables Population, Area, Children, Age, Income, Marital, Gender, ReAdmis, and vitD_levels.

Figure 4

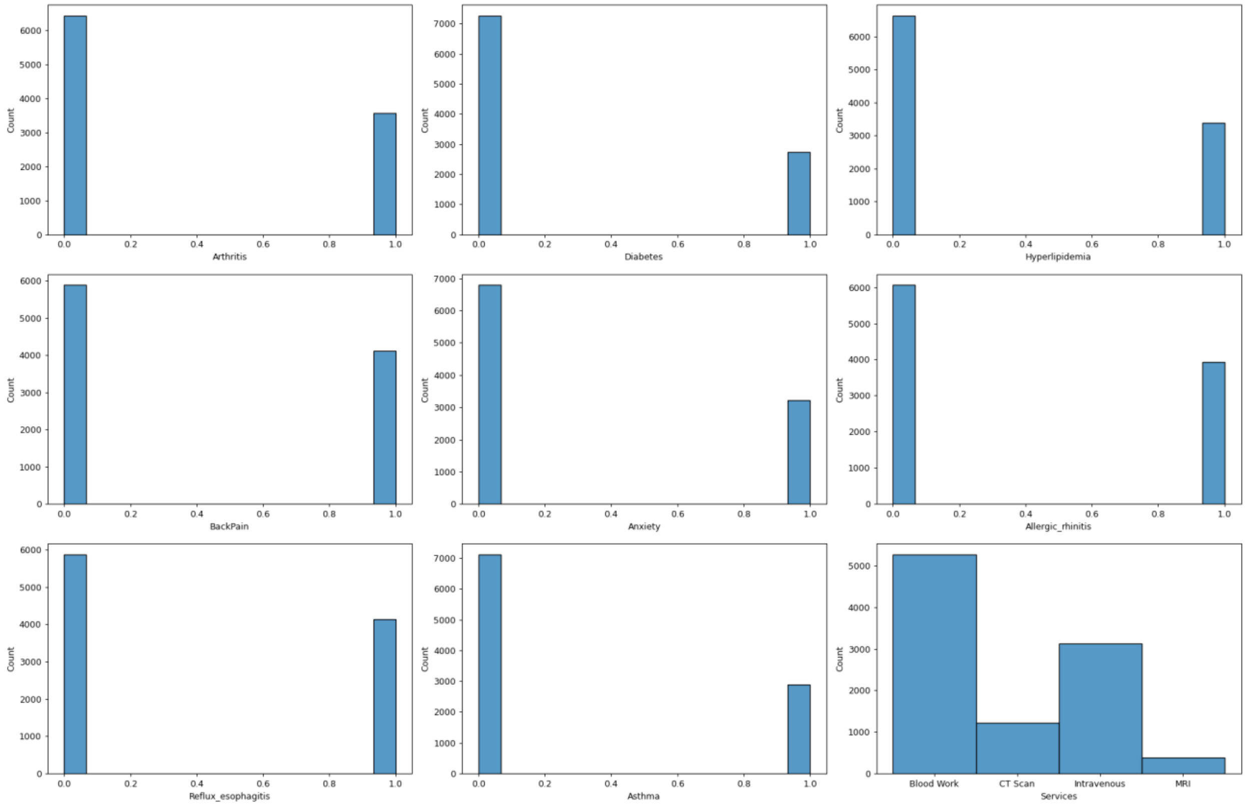
Univariate Visualizations for the Second Set of Variables



Note. This visualization shows the univariate bar charts for variables Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, Complication_risk, and Overweight.

Figure 5

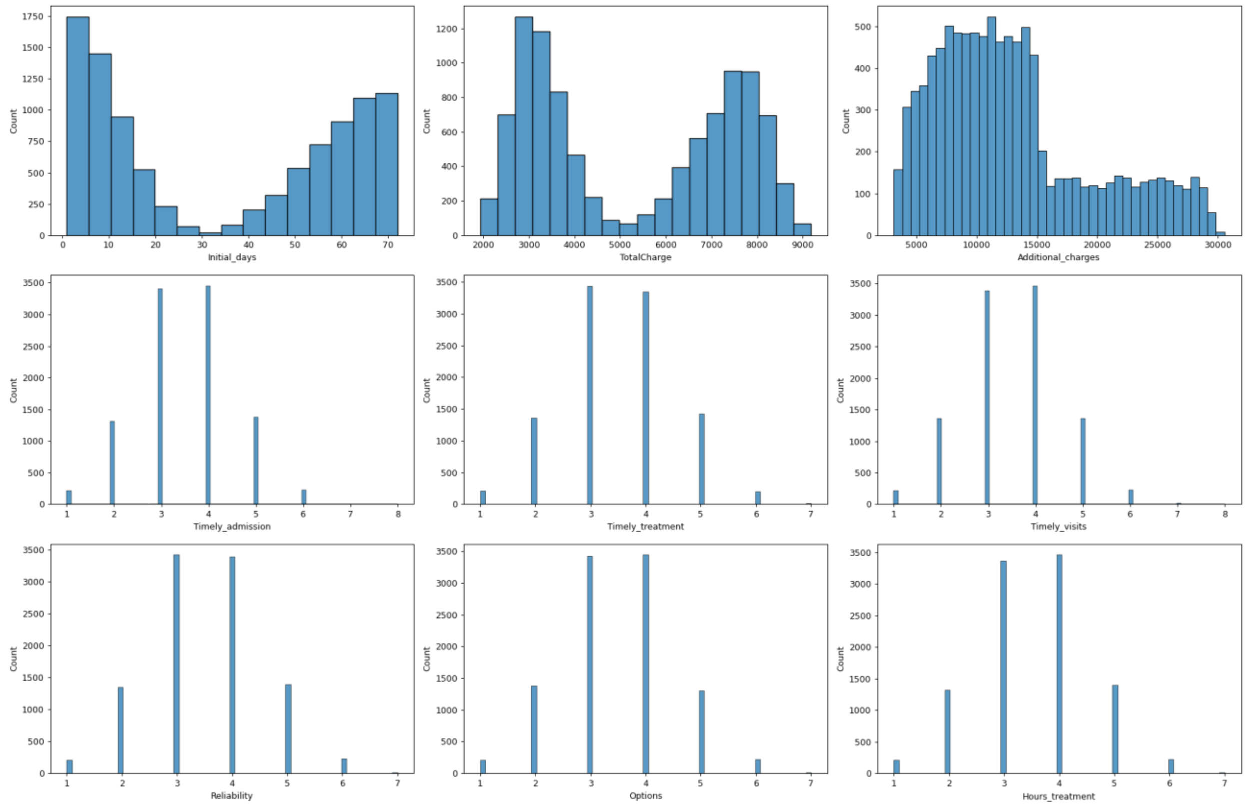
Univariate Visualizations for the Third Set of Variables.



Note. This visualization shows the univariate bar charts for variables Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, and Services.

Figure 6

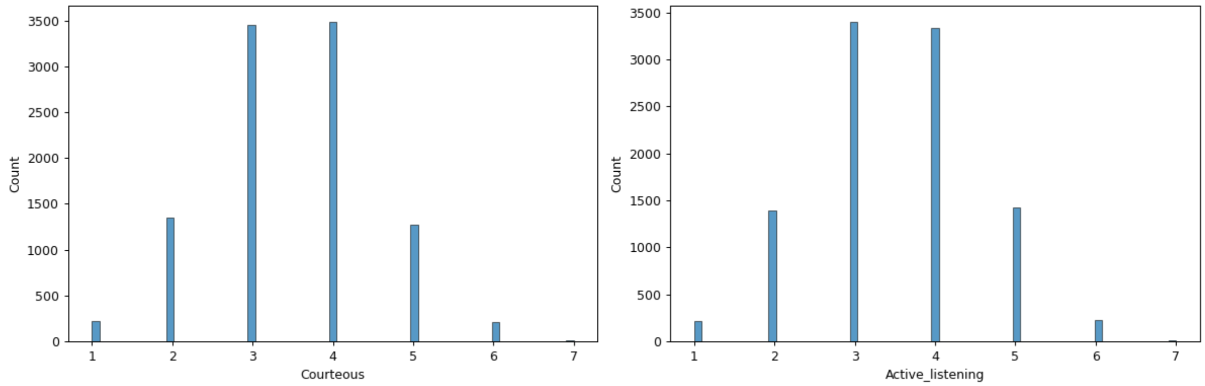
Univariate Visualizations for the Fourth Set of Variables



Note. This visualization shows the univariate bar charts for variables Initial_days, TotalCharge, Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, and Hours_treatment.

Figure 7

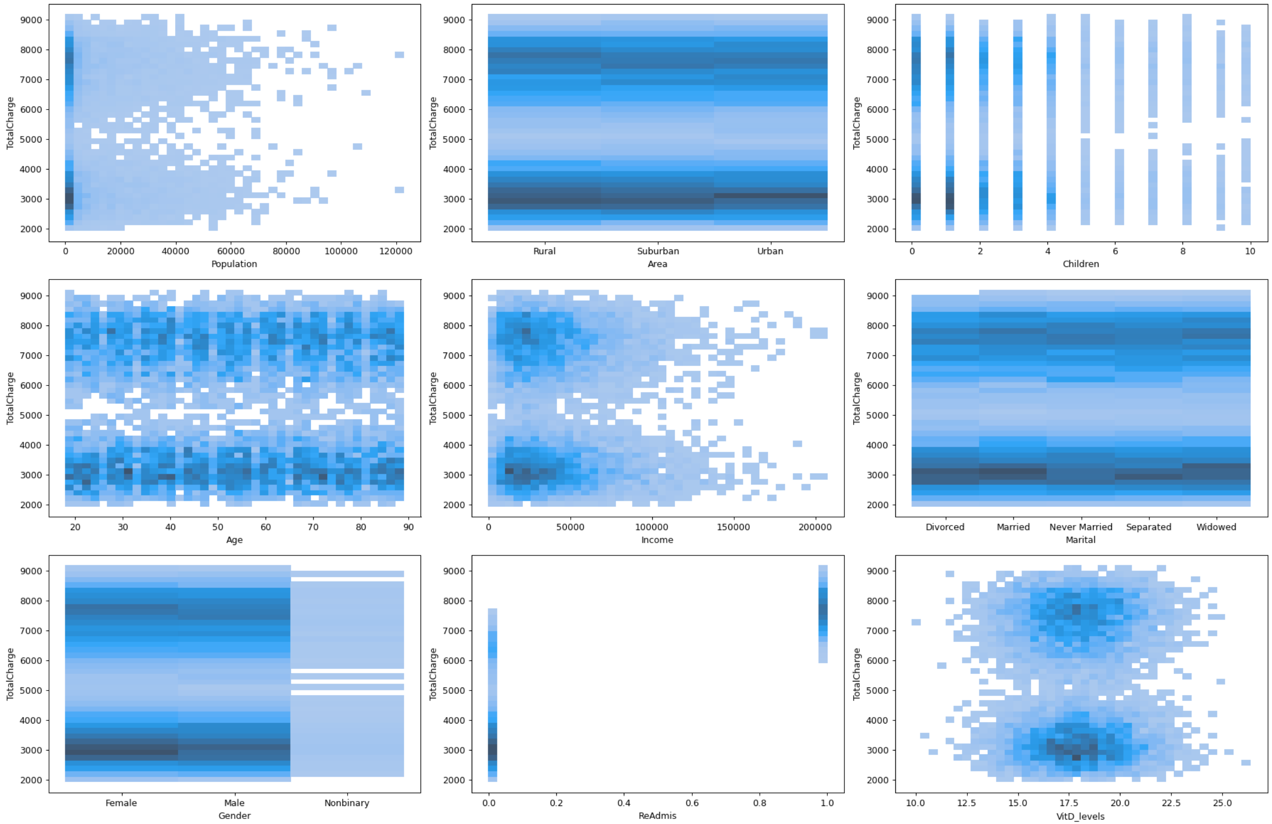
Univariate Visualizations for the Fifth Set of Variables



Note. This visualization shows the univariate bar charts for variables Courteous and Active_listening.

Figure 8

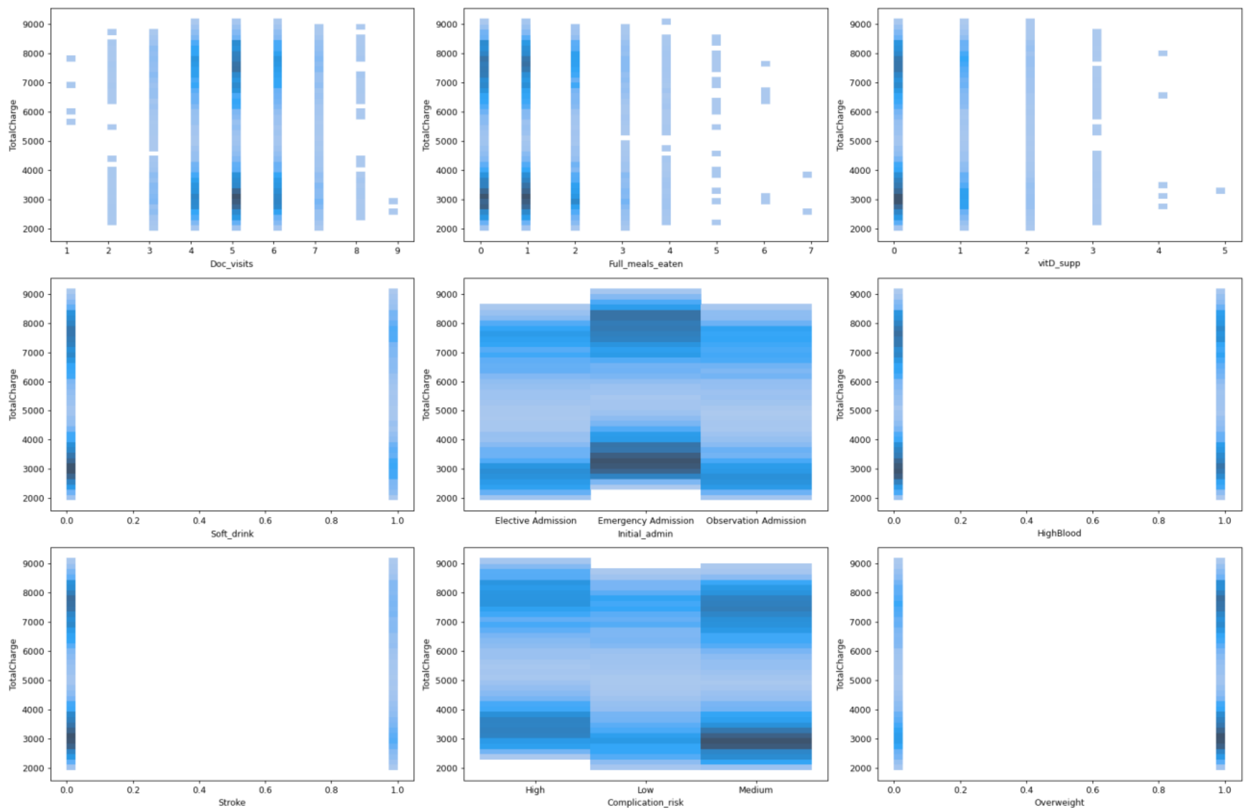
Bivariate Visualization of Population, Area, Children, Age, Income, Marital, Gender, ReAdmis, and vitD_levels



Note. Histograms of the predictor variables Population, Area, Children, Age, Income, Marital, Gender, and vitD_levels by the response variable TotalCharge.

Figure 9

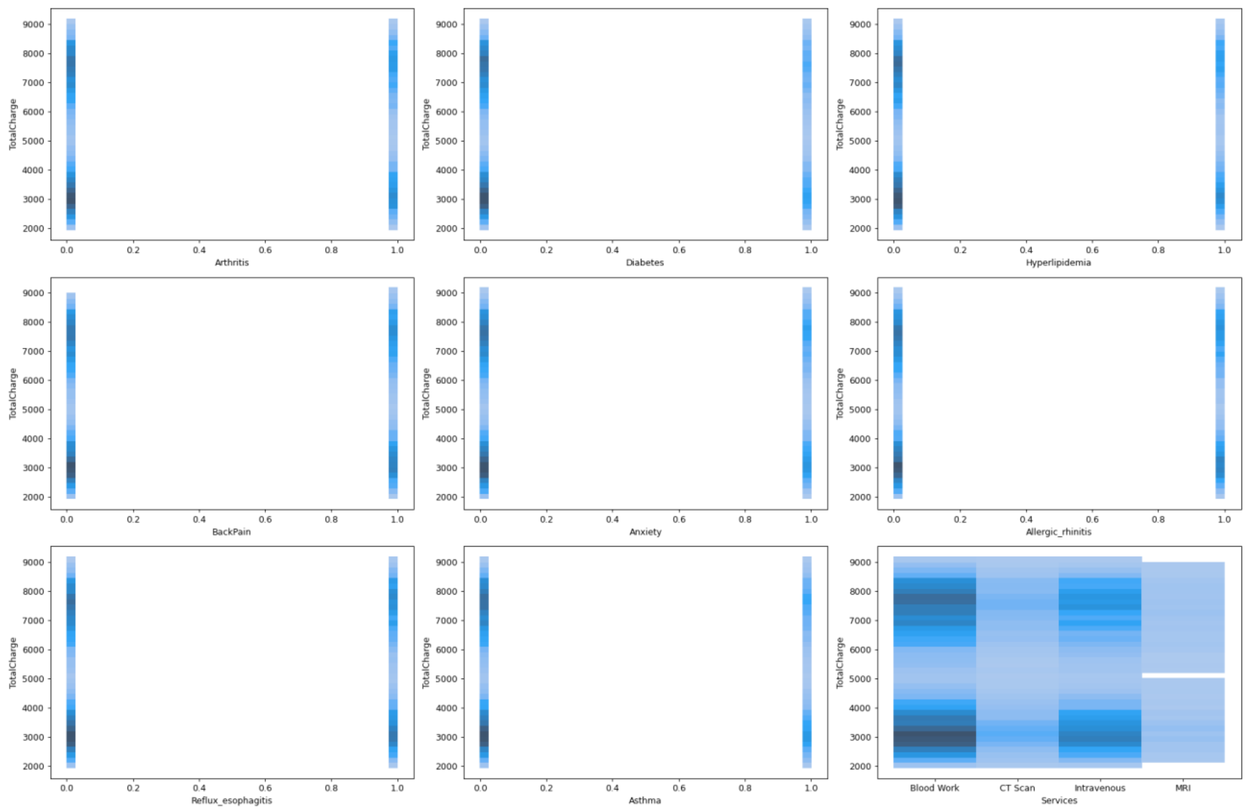
Bivariate Visualization of Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, Complication_risk, and Overweight



Note. Histograms of the predictor variables: Doc_visits, Full_meals_eaten, vitD_supp, Soft_drink, Initial_admin, HighBlood, Stroke, and Complication_risk Overweight by the response variable TotalCharge.

Figure 10

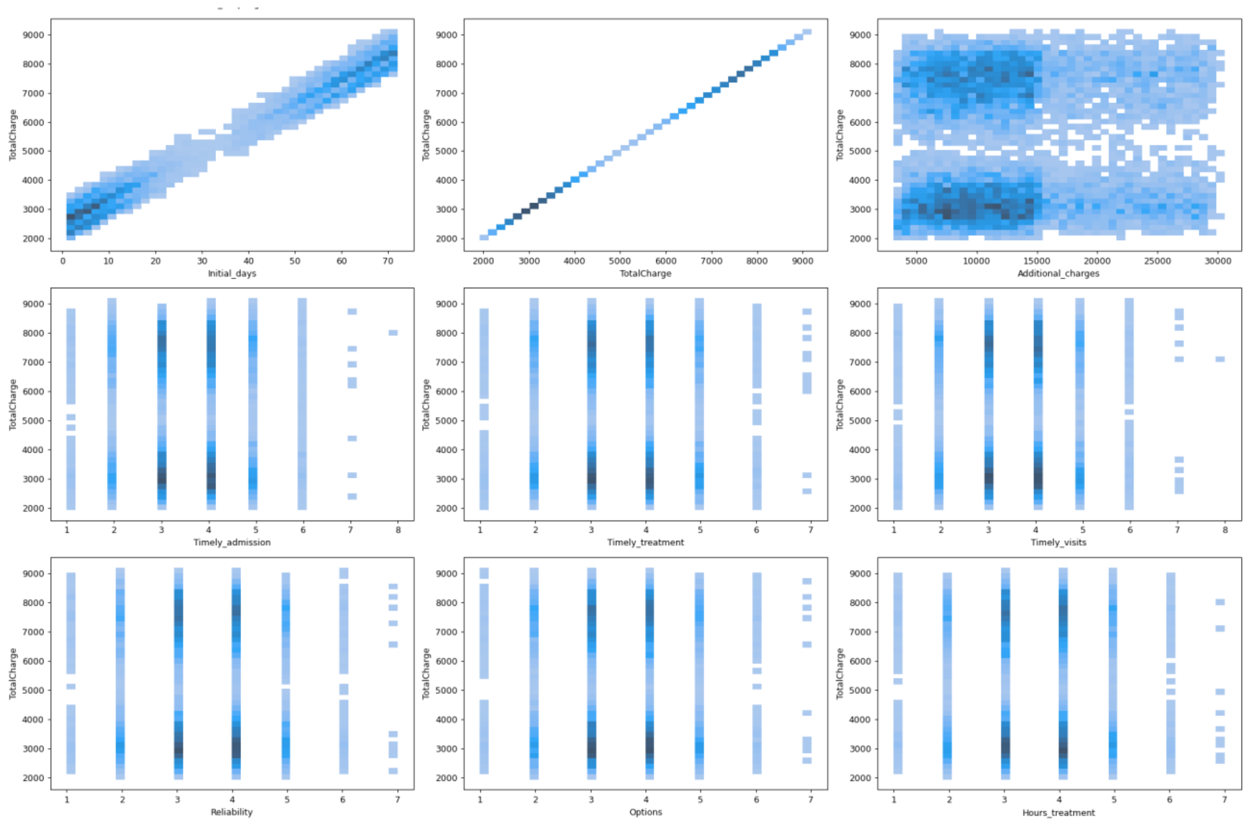
Bivariate Visualization of Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, and Services



Note. Histograms of the predictor variables Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, Asthma, and Services by the response variable TotalCharge.

Figure 11

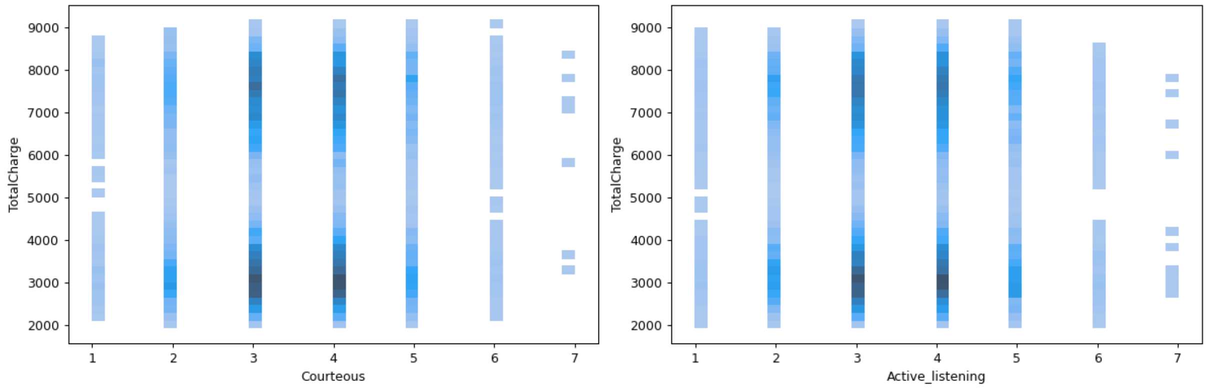
Bivariate Visualization of Initial_days, TotalCharge, Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, and Hours_treatment



Note. Histograms of the predictor variables Initial_days, TotalCharge, Additional_charges, Timely_admission, Timely_treatment, Timely_visits, Reliability, Options, and Hours_treatment by the response variable TotalCharge.

Figure 12

Bivariate Visualization of Courteous and Active_listening



Note. Histograms of the predictor variables Courteous and Active_listening by the response variable TotalCharge.

Figure 13

Generate Dummy Variables and Check Variance Inflation Factor

```
# Create Dummy Variables for all categorical independent variables
med_rec_dummies=pd.get_dummies(med_rec, drop_first=True)
```

```
# Split the data into dependent and independent variables
dependent = med_rec_dummies.TotalCharge
independent = med_rec_dummies.drop(columns=['TotalCharge'])
independent = sm.add_constant(independent)
```

```
# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = independent.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(independent.astype(float).values, i)
                          for i in range(len(independent.columns))]
```

```
# Creates a list of independent variables that have excessive
# VIF numbers, then drops them from the data
feature_list = []
for feature, VIF in zip(vif_data.feature, vif_data.VIF):
    if VIF > 5 and feature != 'const':
        print("Variable: ",feature," VIF: ",VIF)
        feature_list.append(feature)
for item in feature_list:
    if item in independent:
        independent = independent.drop(columns=item)
```

```
Variable:  Age  VIF:  9.285170093623902
Variable:  HighBlood  VIF:  7.87257339132548
Variable:  Additional_charges  VIF:  16.325647750955838
```

Note. Dummy variables are created for the different levels of the categorical variables. The data set is then split between the independent and dependent variables. A DataFrame is then designed to hold the variance inflation factor data(VIF). The generated list is then combed over, looking for a VIF of over 5, subsequently dropped from the independent data set.

Figure 14

Initial Multiple Regression Model

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 2.553e+05
Date:                Tue, 09 Nov 2021   Prob (F-statistic):               0.00
Time:                        10:19:45   Log-Likelihood:                -37975.
No. Observations:                7000   AIC:                         7.604e+04
Df Residuals:                    6956   BIC:                         7.634e+04
Df Model:                          43
Covariance Type:            nonrobust
=========================================================================================================
                                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------------
const                                2313.7881      9.449    244.876      0.000    2295.266    2332.311
Population                           3.586e-05   4.42e-05      0.810      0.418   -5.09e-05       0.000
Children                                0.2693      0.303      0.888      0.375      -0.325       0.864
Income                              -3.151e-06   2.29e-05     -0.138      0.891   -4.81e-05    4.18e-05
ReAdmis                                 3.5443      2.625      1.350      0.177      -1.601       8.689
VitD_levels                             0.1183      0.326      0.363      0.716      -0.520       0.757
Doc_visits                              0.4023      0.628      0.641      0.522      -0.828       1.633
Full_meals_eaten                        0.4878      0.666      0.733      0.464      -0.817       1.792
vitD_supp                               0.9748      1.053      0.926      0.355      -1.089       3.039
Soft_drink                             -1.2459      1.501     -0.830      0.407      -4.188       1.696
Stroke                                  2.7897      1.651      1.690      0.091      -0.446       6.025
Overweight                              4.2172      1.449      2.910      0.004       1.376       7.058
Arthritis                              74.5315      1.382     53.935      0.000      71.823      77.240
Diabetes                               76.5684      1.482     51.676      0.000      73.664      79.473
Hyperlipidemia                         94.2474      1.394     67.597      0.000      91.514      96.981
BackPain                               86.4051      1.343     64.346      0.000      83.773      89.037
Anxiety                                87.2675      1.420     61.452      0.000      84.484      90.051
Allergic_rhinitis                      60.6421      1.353     44.825      0.000      57.990      63.294
Reflux_esophagitis                     59.7946      1.340     44.612      0.000      57.167      62.422

Asthma                                  0.4837      1.456      0.332      0.740      -2.370       3.338
Initial_days                           81.8635      0.048   1700.199      0.000      81.769      81.958
Timely_admission                       -0.2418      0.947     -0.255      0.799      -2.098       1.615
Timely_treatment                        0.2153      0.876      0.246      0.806      -1.502       1.932
Timely_visits                          -1.3033      0.816     -1.598      0.110      -2.902       0.296
Reliability                            -0.9607      0.726     -1.324      0.186      -2.383       0.462
Options                                -0.9828      0.757     -1.299      0.194      -2.466       0.501
Hours_treatment                         0.4258      0.781      0.545      0.586      -1.106       1.957
Courteous                               0.2472      0.742      0.333      0.739      -1.207       1.702
Active_listening                       -0.0019      0.700     -0.003      0.998      -1.375       1.371
Area_Suburban                           1.4839      1.614      0.920      0.358      -1.679       4.647
Area_Urban                              0.4887      1.617      0.302      0.762      -2.681       3.658
Marital_Married                         0.2858      2.100      0.136      0.892      -3.831       4.403
Marital_Never Married                  -2.5974      2.106     -1.233      0.217      -6.725       1.531
Marital_Separated                      -2.3445      2.101     -1.116      0.264      -6.462       1.773
Marital_Widowed                        -3.6712      2.091     -1.756      0.079      -7.769       0.427
Gender_Male                             0.0958      1.336      0.072      0.943      -2.522       2.714
Gender_Nonbinary                        4.0872      4.622      0.884      0.377      -4.973      13.147
Initial_admin_Emergency Admission     512.8090      1.624    315.840      0.000     509.626     515.992
Initial_admin_Observation Admission     0.7223      1.882      0.384      0.701      -2.966       4.411
Complication_risk_Low                -416.8517      1.849   -225.390      0.000    -420.477    -413.226
Complication_risk_Medium             -412.7453      1.498   -275.458      0.000    -415.683    -409.808
Services_CT Scan                        1.8551      2.105      0.881      0.378      -2.271       5.981
Services_Intravenous                    0.4632      1.493      0.310      0.756      -2.463       3.389
Services_MRI                            1.8085      3.470      0.521      0.602      -4.994       8.611
==============================================================================
Omnibus:                    27401.493   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1144.755
Skew:                           0.388   Prob(JB):                    2.63e-249
Kurtosis:                       1.177   Cond. No.                     7.29e+05
==============================================================================
```

Note. This is the summary of the original multiple regression model.

Figure 15

Root Mean Square Error of the Original Model

```
# Creates a list of predictions
yhat = result.predict(x_test)

# Root Mean Squared Error (RMSE)
rmse = math.sqrt(metrics.mean_squared_error(y_test, yhat))
adj_rmse = rmse/(statistics.mean(y_test))

# prints the RMSE and RMSE as a percentage of
# the mean of observations
print('RMSE: ', rmse)
print('Adjust RMSE: ', adj_rmse)
```

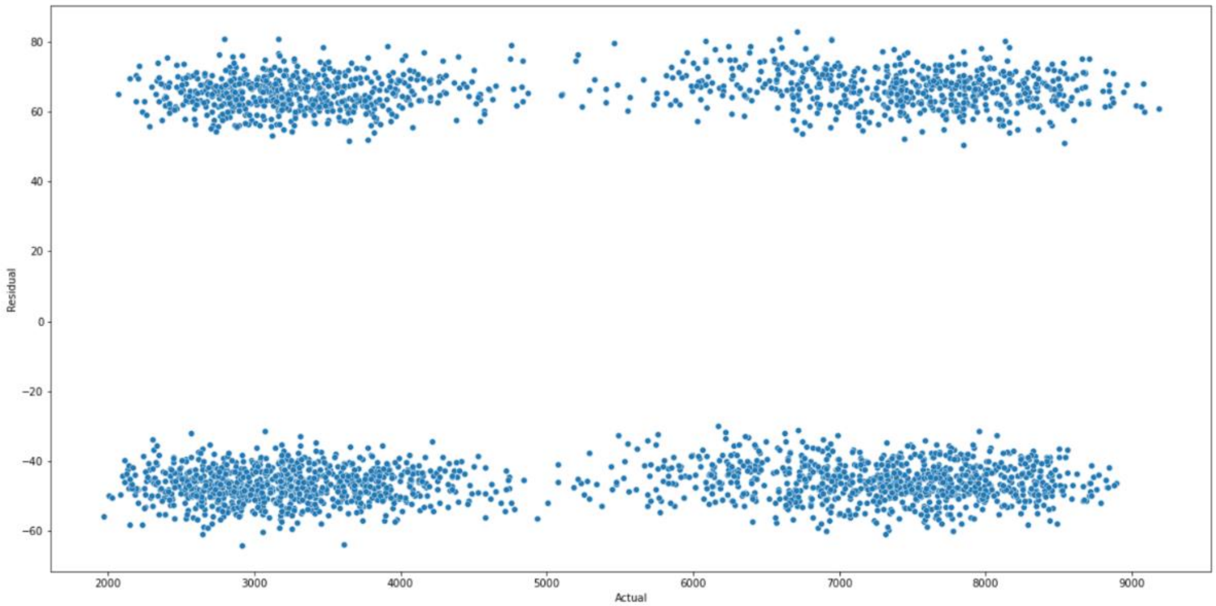
```
RMSE:  55.49312180479003
Adjust RMSE:  0.010376341834336271
```

Note. The code uses the original model to make predictions based on the set of x_test inputs. The predictions are used in the calculation and the actual values of y_test to give the Root Mean Square Error (RMSE). This is then divided by the mean of the observations to determine the RMSE as a percentage.

Figure 16

Residuals from the Original Model

```
fig = plt.figure(figsize=(20,10))
res = pd.DataFrame()
res['Residual'] = y_test-yhat
res['Actual'] = y_test
sns.scatterplot(data=res,y='Residual', x='Actual')
plt.show()
```



Note. This scatterplot shows the residuals against the actual values along the original model's regression line (0).

Figure 17

Feature Selection using Backward Elimination with Adjust R-Squared

```
# Global variables to perfrom checks
sig_features = []
highest = 'no'
last_highest = 'yes'
adj_check = independent
current_adj = result.rsquared_adj


# Removes features with the highest p-values, then evaluates the Adjust R-Squared
# in order to determine if the feature helped predictability
elim_result = sm.OLS(dependent, independent.astype(float)).fit(disp=0)
min_adj = elim_result.rsquared_adj
while highest != last_highest:
    max_pvalue = 0.05
    last_highest = highest

    if min_adj < current_adj and highest != 'no':
        min_adj = current_adj
        adj_check = adj_check.drop(columns=highest)
    elif min_adj > current_adj and highest != 'no':
        sig_features.append(highest)
        independent = adj_check

    for feature, pvalue in zip(independent.columns, elim_result.pvalues):
        if pvalue > max_pvalue and feature not in sig_features:
            max_pvalue = pvalue
            highest = feature
    if highest != last_highest:
        independent = independent.drop(columns=highest)
        elim_result = sm.OLS(dependent, independent.astype(float)).fit(disp=0)
        current_adj = elim_result.rsquared_adj
```

Note. This code uses backward elimination to remove the features with the highest p-values. It then compares the Adjusted R-Square of the model without the feature to the previous model's Adjusted R-Square, leaving the feature out or replacing the feature depending on which model has the higher Adjusted R-Square. It then moves on to the next feature until it has removed the features required to achieve the highest Adjusted R-Square.

Figure 18

Reduced Multiple Regression Model

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 4.356e+05
Date:                Tue, 09 Nov 2021   Prob (F-statistic):               0.00
Time:                        10:19:51   Log-Likelihood:                -37992.
No. Observations:                7000   AIC:                         7.604e+04
Df Residuals:                    6974   BIC:                         7.621e+04
Df Model:                          25
Covariance Type:            nonrobust
=================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
const                          2322.7632      5.964    389.492      0.000    2311.073    2334.454
ReAdmis                           1.1247      2.607      0.431      0.666      -3.986       6.235
Full_meals_eaten                  0.6421      0.654      0.982      0.326      -0.640       1.924
Overweight                        2.7114      1.455      1.864      0.062      -0.141       5.563
Arthritis                        73.3018      1.380     53.128      0.000      70.597      76.006
Diabetes                         73.9374      1.490     49.607      0.000      71.016      76.859
Hyperlipidemia                   92.9635      1.391     66.811      0.000      90.236      95.691
BackPain                         86.6803      1.344     64.506      0.000      84.046      89.314
Anxiety                          87.1214      1.419     61.415      0.000      84.341      89.902
Allergic_rhinitis                61.2799      1.349     45.442      0.000      58.636      63.923
Reflux_esophagitis               60.1670      1.340     44.901      0.000      57.540      62.794
Initial_days                     81.9213      0.048   1709.877      0.000      81.827      82.015
Timely_visits                    -0.8579      0.659     -1.302      0.193      -2.150       0.434
Reliability                      -0.8647      0.717     -1.207      0.228      -2.269       0.540
Options                          -1.7377      0.730     -2.381      0.017      -3.168      -0.307
Courteous                         1.0217      0.693      1.475      0.140      -0.336       2.380
Area_Suburban                     2.2043      1.616      1.364      0.173      -0.963       5.372
Area_Urban                        1.5298      1.618      0.946      0.344      -1.641       4.701
Marital_Never Married            -4.7707      1.820     -2.621      0.009      -8.339      -1.202

Marital_Separated                -1.5957      1.825     -0.874      0.382      -5.174       1.983
Marital_Widowed                  -4.3146      1.777     -2.428      0.015      -7.798      -0.831
Gender_Nonbinary                  1.7347      4.432      0.391      0.696      -6.954      10.424
Initial_admin_Emergency Admission   509.7760      1.324    384.959      0.000     507.180     512.372
Complication_risk_Low          -416.5621      1.839   -226.476      0.000    -420.168    -412.957
Complication_risk_Medium       -413.1967      1.504   -274.812      0.000    -416.144    -410.249
Services_CT Scan                 -0.5131      2.052     -0.250      0.803      -4.535       3.509
==============================================================================
Omnibus:                    26986.816   Durbin-Watson:                   2.018
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1146.980
Skew:                           0.368   Prob(JB):                    8.64e-250
Kurtosis:                       1.159   Cond. No.                         404.
==============================================================================
```

Note. The code and summary for the reduced multiple regression model after the appropriate features have been selected.

Figure 19

Root Mean Square Error of the Reduced Model

```
# Creates a list of predictions
yhat = result.predict(x_test)

# Root Mean Squared Error (RMSE)
rmse = math.sqrt(metrics.mean_squared_error(y_test, yhat))
adj_rmse = rmse/(statistics.mean(y_test))

# prints the RMSE and RMSE as a percentage of
# the mean of observations
print('RMSE: ', rmse)
print('Adjust RMSE: ', adj_rmse)
```

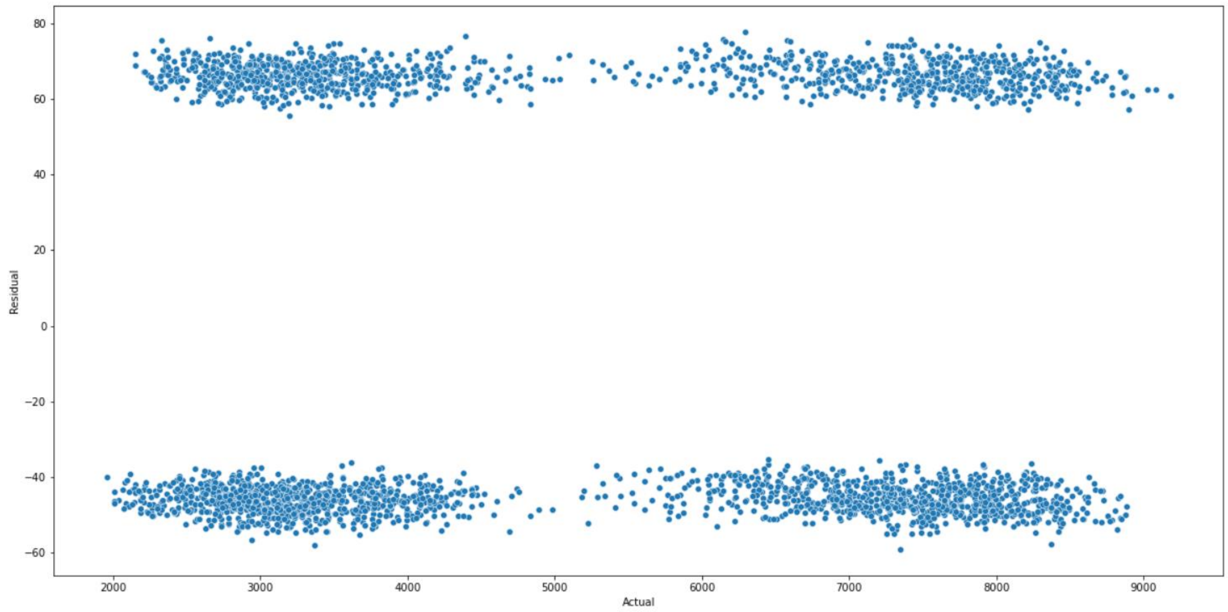
```
RMSE:  55.16546867789881
Adjust RMSE:  0.010469330330034549
```

Note. The code uses the original model to make predictions based on the set of x_test inputs. The predictions are used in the calculation and the actual values of y_est to give the Root Mean Square Error (RMSE) for the reduced model. This is then divided by the mean of the observations to determine the RMSE as a percentage.

Figure 20

Residuals from the Reduced Model

```
fig = plt.figure(figsize=(20,10))
res = pd.DataFrame()
res['Residual'] = y_test-yhat
res['Actual'] = y_test
sns.scatterplot(data=res,y='Residual', x='Actual')
plt.show()
```



Note. This scatterplot shows the residuals against the actual values along the reduced model's regression line (0).

Figure 21

Predictions Using the Reduced Model

```
pred_act = pd.DataFrame()
pred_act['Prediction'] = yhat
pred_act['Actual'] = y_test
pred_act.head(10)
```

	Prediction	Actual
6236	7886.46612	7845.323000
1150	2183.897859	2138.895704
8565	8481.643206	8437.948000
4846	4483.466656	4437.368923
6794	7441.641109	7504.746000
8861	7254.153001	7212.280000
6064	7988.229881	7941.052000
405	2973.202765	3036.630049
9306	6137.296871	6089.776000
4620	4327.849836	4281.832098

Note. This code creates a DataFrame to hold the predictions from the reduced model alongside the actual values from the y_test data set. The first ten results in the DataFrame are shown to explain how the predictions line up with the actual values.

Figure 22

Coefficients of the Variables for the Multiple Regression Function

```
result.params

const                               2322.763228
ReAdmis                                1.124706
Full_meals_eaten                       0.642060
Overweight                             2.711412
Arthritis                             73.301808
Diabetes                              73.937356
Hyperlipidemia                        92.963507
BackPain                              86.680283
Anxiety                               87.121410
Allergic_rhinitis                     61.279907
Reflux_esophagitis                    60.166964
Initial_days                          81.921306
Timely_visits                         -0.857919
Reliability                           -0.864716
Options                               -1.737672
Courteous                              1.021727
Area_Suburban                          2.204310
Area_Urban                             1.529769
Marital_Never Married                 -4.770695
Marital_Separated                     -1.595662
Marital_Widowed                       -4.314608
Gender_Nonbinary                       1.734703
Initial_admin_Emergency Admission    509.775968
Complication_risk_Low               -416.562140
Complication_risk_Medium            -413.196669
Services_CT Scan                      -0.513080
dtype: float64
```

Note. These are the coefficients to the variables in the reduced model for the multiple regression equation.