**D208 Task 2: Logistic 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 whether they will need to be readmitted within a month of release (‘ReAdmis’ recorded as ‘Yes’) and can this be predicted with a logistic regression model?”

**A.2. Goals**

This project aims to find the most influential variables affecting whether patients will be readmitted within 30 days of their initial stay and to build a logistic regression model to predict that outcome.

These insights can assist hospital administrators and analysts understand what factors drive patients needing to be readmitted shortly after leaving the hospital to make the necessary changes for greater efficiency and better patient health outcomes.

**B.1. Summary of assumptions**

Logistic regression models are assumed to (Bobbitt, 2020 [1]):

1. Have independent variables with low correlations between one another (no strong multicollinearity).
2. The independent variables have a linear relationship with the Logit of the dependent variable’s probability , defined as . That is, where are the coefficients of the logistic regression model and the are the independent variable values for a given observation.
3. The target variable is a binary categorical value (e.g. “Yes” or “No”, 0 or 1).
4. The observations (data entry rows) are independent of one another.

**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 logistic regression modeling and variance inflation factor calculations. From scikit-learn, various packages were used in the recursive feature elimination procedure and calculations of confusion matrices and accuracy scores.

**B.3. Appropriate technique**

The goal of this project looks to predict a binary outcome (‘Yes’ or ‘No’ to the categorical column ‘ReAdmis’) from numerical and (initially) categorical independent variables, where the categorical predictor variables will be one-hot encoded into integer values. To meet the assumptions of the model, the independent variables will be selected to ensure limited multicollinearity and a linear relationship with the Logit of the target variable. Since the observations are independent of one another and the target variable is binary, logistic regression is a suitable approach for a predictive model.

**C.1. Data cleaning goals**

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 “logistic.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 ‘ReAdmis’. Categorical variables additionally include .value\_counts()).

relevant\_columns = ['ReAdmis', 'Initial\_days', 'Children', 'Age', 'Population', 'Lat', 'Additional\_charges',  
 'Complication\_risk', 'Initial\_admin', 'BackPain', 'Arthritis', 'Anxiety', 'Asthma',  
 'Hyperlipidemia', 'Stroke', 'Overweight', 'Services']  
inspect\_data(df\_med[relevant\_columns])

count 10000

unique 2

top No

freq 6331

Name: ReAdmis, dtype: object

ReAdmis

No 6331

Yes 3669

Name: count, dtype: int64

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: 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: 8588

count 10000.000000

mean 38.751099

std 5.403085

min 17.967190

25% 35.255120

50% 39.419355

75% 42.044175

max 70.560990

Name: Lat, 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

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 7107

Name: Asthma, dtype: object

Asthma

No 7107

Yes 2893

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 8007

Name: Stroke, dtype: object

Stroke

No 8007

Yes 1993

Name: count, dtype: int64

count 10000

unique 2

top Yes

freq 7094

Name: Overweight, dtype: object

Overweight

Yes 7094

No 2906

Name: count, dtype: int64

count 10000

unique 4

top Blood Work

freq 5265

Name: Services, dtype: object

Services

Blood Work 5265

Intravenous 3130

CT Scan 1225

MRI 380

Name: count, dtype: int64

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

* 'Initial\_days'
* 'Children'
* ‘Age’
* ‘Population’
* ‘Lat’
* 'Additional\_charges'

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

* 'Complication\_risk'
* 'Initial\_admin'
* 'BackPain'
* 'Arthritis'
* 'Anxiety'
* 'Asthma’
* 'Hyperlipidemia'
* ‘Stroke’
* ‘Overweight’
* ‘Services’

These initial variables were selected by looking at the correlation matrix values restricted to the target variable ‘ReAdmis’ and those indicating serious health complications with a patient.

Approximately 37% of patients need to be readmitted within a month of leaving the hospital. The length of initial stay is a bimodal distribution clustered into patients staying less than 30 days and those who stay longer than 30 days (with a maximum of 72). Around 80% of patients are regarded as having a complication risk of medium or high, with just over 50% of admissions being for emergencies.

The binary health conditions (e.g. ‘Asthma’, ‘Anxiety’) generally split with affecting ~30-40% of patients, with being overweight a notable exception affecting 70% of patients.

**C.3. Visualizations**

A graph with blue rectangles

Description automatically generated

A graph with blue bars

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated

A graph of age and age

Description automatically generated

A graph with blue bars

Description automatically generated

A graph with a blue line

Description automatically generated with medium confidence

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 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 bars

Description automatically generated

A graph with blue rectangles

Description automatically generated

A graph of services with blue bars

Description automatically generated with medium confidence

A graph of a bivariate density plot

Description automatically generated

A graph of bivariate histograms

Description automatically generated

A graph of bivariate density plot of age

Description automatically generated

A graph of a bivariate density plot

Description automatically generated

A graph of a normal distribution

Description automatically generated

A graph of a number of charges

Description automatically generated

A graph of different colored bars

Description automatically generated

A graph of a bar chart

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph with blue and orange bars

Description automatically generated

A graph of a bar graph

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 logistic regression model, the only transformations necessary are one-hot encoding on categorical columns (as logistic 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. ‘Asthma’ becomes ‘Asthma\_Yes’ with a ‘No’ value for ‘Asthma’ corresponding to 0 in the ‘Asthma\_Yes’ column). Creating a column of 1s was done simply through df\_med['model\_constant'] = 1. See the attached file “logistic.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("\nVerifying new columns have expected values:")  
print(df\_med['Diabetes\_Yes'].value\_counts())  
print(df\_med['Initial\_admin\_Emergency Admission'].value\_counts())  
print(f"\nColumns encoded: {global\_encoded\_columns}\n")

# Creating a vector of 1s for a constant term.  
df\_med['model\_constant'] = 1

**C.5. Prepared data set**

See attached file “medical\_transformed\_logistic.csv”.

**D.1. Initial model**

Initial model created from:

# Initial model independent variables  
model\_indp\_var = ['model\_constant', 'Initial\_days', 'Children', 'Age', 'Population', 'Lat', 'Additional\_charges',  
 'Complication\_risk\_Medium', 'Complication\_risk\_Low', 'Initial\_admin\_Emergency Admission',  
 'Initial\_admin\_Observation Admission', 'BackPain\_Yes', 'Arthritis\_Yes', 'Anxiety\_Yes', 'Asthma\_Yes',  
 'Hyperlipidemia\_Yes', 'Stroke\_Yes', 'Overweight\_Yes', 'Services\_MRI', 'Services\_CT Scan']  
model\_dep\_var = 'ReAdmis\_Yes'  
X\_0 = df\_med[model\_indp\_var]  
y\_0 = df\_med[model\_dep\_var]  
  
  
# Logistic regression model for target y ('ReAdmis\_Yes' in this case) with independent variables X (see model\_indp\_var)  
def log\_model(y, X):  
 model = sm.Logit(y, X)  
 results = model.fit()  
 print(results.summary())  
 print(f"\nAIC: {2 \* (2 + results.df\_model) - 2 \* results.llf}")  
 print(f"Predicted percentage of readmissions: "  
 f"{np.round(results.predict()[results.predict() >= 0.5].sum() / len(results.predict()), 4)}")  
 print(f"Accuracy: {accuracy\_score(y, np.round(results.predict()))}\n")  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=5)  
 model\_sk = LogisticRegression(C=1.0, class\_weight='balanced', max\_iter=200, penalty='l1', solver='liblinear', random\_state=5)  
 model\_sk.fit(X\_train, y\_train)  
 print(f"Coefficients: {dict(zip(X.columns, model\_sk.coef\_[0]))}\n")  
 y\_pred = model\_sk.predict(X\_test)  
 conf\_matrix = confusion\_matrix(y\_test, y\_pred)  
 validation = cross\_val\_score(model\_sk, X\_train, y\_train, cv=8)  
 metric\_scores = classification\_report(y\_test, y\_pred)  
 print(f"Confusion matrix: \n{conf\_matrix}\n")  
 print(f"Cross validation scores: {validation}\n")  
 print(f"Classification report: \n{metric\_scores}\n")  
  
 sns.heatmap(X.corr(), annot=True)  
 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

Description automatically generated

model\_sk (created from scikit-learn’s LogisticRegression()) coefficients:

Coefficients: {'model\_constant': -28.850533168509347, 'Initial\_days': 1.0632617563176672, 'Children': 0.05894213696322496, 'Age': -0.01142960365434303, 'Population': 7.1890883369318205e-06, 'Lat': -0.007046354576962525, 'Additional\_charges': 5.132044917364805e-05, 'Complication\_risk\_Medium': -0.11465762192023272, 'Complication\_risk\_Low': -1.127620071184741, 'Initial\_admin\_Emergency Admission': 1.6621318999178265, 'Initial\_admin\_Observation Admission': 0.3240922873304823, 'BackPain\_Yes': 0.1347185882131392, 'Arthritis\_Yes': -0.747731946217823, 'Anxiety\_Yes': -0.692967402316351, 'Asthma\_Yes': -0.736599466935999, 'Hyperlipidemia\_Yes': 0.0496881352911525, 'Stroke\_Yes': 0.8633403530006591, 'Overweight\_Yes': -0.22543266767472403, 'Services\_MRI': 1.7328744071728586, 'Services\_CT Scan': 1.2314546664691792}

A screenshot of a computer program

Description automatically generated

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

This initial model has incredibly high accuracy of 0.9862 and a log-likelihood of -335.99. While this is promising, the number of variables used is excessive and many of them have low z-scores with standard errors approaching or exceeding the magnitude of their coefficients. Excluding the constant term, variance inflation factors are all below 5.

A screenshot of a computer program

Description automatically generated

As variance inflation provides no guidance, recursive feature elimination (RFE) recursively eliminates features from the model using a regression function, removing the least important feature at each step. In this way, we can iteratively reduce the model and select a reduced set of features providing the best model accuracy and AIC. Initially, it will be set to 12 features to identify the seven least important of the starting 19 independent variables (excluding the constant). Reviewing the initial model, there are seven variables with p-values above 0.05, so it’s suspected their removal may produce a refined model with optimal accuracy and/or AIC. This requires confirmation, so in principle, the optimal reduced model may have fewer or more than 12 independent variables, pending comparisons of accuracy, AIC, and the statistical significance of the remaining independent variables. However, after significant comparison and tuning, it was discovered reductions recommended by RFE with more than 12 features produced inferior results.

Performing recursive feature elimination to retain 12 features:

Index(['Initial\_days', 'Population', 'Additional\_charges',

'Complication\_risk\_Low', 'Initial\_admin\_Emergency Admission',

'Initial\_admin\_Observation Admission', 'Arthritis\_Yes', 'Anxiety\_Yes',

'Asthma\_Yes', 'Stroke\_Yes', 'Services\_MRI', 'Services\_CT Scan'],

dtype='object')

{'model\_constant': 9, 'Initial\_days': 1, 'Children': 3, 'Age': 2, 'Population': 1, 'Lat': 8, 'Additional\_charges': 1, 'Complication\_risk\_Medium': 7, 'Complication\_risk\_Low': 1, 'Initial\_admin\_Emergency Admission': 1, 'Initial\_admin\_Observation Admission': 1, 'BackPain\_Yes': 4, 'Arthritis\_Yes': 1, 'Anxiety\_Yes': 1, 'Asthma\_Yes': 1, 'Hyperlipidemia\_Yes': 6, 'Stroke\_Yes': 1, 'Overweight\_Yes': 5, 'Services\_MRI': 1, 'Services\_CT Scan': 1}

Removing the constant term produces inferior, less accurate models, so the recommendations of less relevant features from RFE are:

* ‘Lat’
* ‘Complication\_risk\_Medium’
* ‘Hyperlipidemia\_Yes’
* ‘Overweight\_Yes’
* ‘BackPain\_Yes’
* ‘Children’
* ‘Age’

As it so happens, the p-values above 0.05 in the model’s summary are ‘Overweight\_Yes’, ‘Hyperlipidemia\_Yes’, ‘BackPain\_Yes’, ‘Complication\_risk\_Medium’, ‘Lat’, ‘Population’, and ‘Children’. Removing ‘Age’ and ‘Complication\_risk\_Medium’ ultimately produces inferior accuracies and AIC values relative to the initial model. Similarly, ‘Population’ has a standard error on the order of its coefficient, a low z-score, and a high p-value, so we find more accurate models with its removal.

Attempts to remove additional variables led to marginally lower AIC values and accuracy scores, so removing the six variables ‘Lat’, ‘Hyperlipidemia\_Yes’, ‘Overweight\_Yes’, ‘BackPain\_Yes’, ‘Children’, and ‘Population’ is the preferred course of action in producing the most accurate possible refinement of the initial model.

**D.3. Reduced logistic regression model**

The resulting reduced model:

A screenshot of a computer

Description automatically generated

Coefficients: {'model\_constant': -28.127693717946993, 'Initial\_days': 1.0430636121959331, 'Age': -0.011217968270747003, 'Additional\_charges': 5.1771629569444486e-05, 'Complication\_risk\_Low': -1.1298514433067528, 'Complication\_risk\_Medium': -0.1096134445549298, 'Initial\_admin\_Emergency Admission': 1.61418785048549, 'Initial\_admin\_Observation Admission': 0.29696205269348575, 'Arthritis\_Yes': -0.7374114475923127, 'Anxiety\_Yes': -0.6462197501417167, 'Asthma\_Yes': -0.751443073970871, 'Stroke\_Yes': 0.8513355459323428, 'Services\_MRI': 1.7353097296628897, 'Services\_CT Scan': 1.1997311876042827}

A screenshot of a computer

Description automatically generated

**E.1. Model comparison**

The initial model had a fairly high accuracy of 0.9862 and pseudo R2 of 0.9489 largely due to the strong correlation between ‘Initial\_days’ and ‘ReAdmis’. The LLR p-value is 0.000, indicating high statistical significance, so from the outset there doesn’t seem to be much room for improvement in the model.

However, as discussed in section D.2., many of the variables had high p-values, high standard errors (relative to the coefficient values), and low z-scores, indicating low statistical significance to predicting whether a patient will need to be readmitted. Removing six variables reduced the AIC from 713.98 to 712.77 and increased the accuracy from 0.9862 to 0.9863. Although the improvements are marginal, the refined model benefits from excluding irrelevant variables and being easier to manage. The z-scores of the remaining variables are all nearly of magnitude 3.0 and above with p-values well below 0.05.

**E.2. Output and calculations**

Refined model accuracy score:

A screen shot of a computer

Description automatically generated

Refined model confusion matrix:

A screenshot of a computer

Description automatically generated

**E.3. Code**

See attached file “logistic.py”.

**F.1. Results**

The logistic regression model produces the following equation:

*= 1.4359 \* 'Initial\_days' – 0.0184 \* ‘Age’ + 0.00008943 \* ‘Additional\_charges’ – 1.7796 \* ‘Complication\_risk\_Medium’ – 0.3383 \* ‘Complication\_risk\_Medium’ + 2.3699 \* ‘Initial\_admin\_Emergency Admission’ + 0.7622 \* ‘Initial\_admin\_Observation Admission’ – 1.2690 \* ‘Arthritis\_Yes’ – 0.9623 \* ‘Anxiety\_Yes’ – 1.3124 \* ‘Asthma\_Yes’ + 1.5058 \* ‘Stroke\_Yes’ + 2.6613 \* ‘Services\_MRI’ + 1.5524 \* ‘Services\_CT Scan’ – 78.4120*

In words, it indicates a patient’s odds ratio of being readmitted will be multiplied by a factor of:

* 4.203 for every additional day in ‘Initial\_days’ (or ~320% increase)
* 0.9818 for every additional year in ‘Age’ (~1.82% decrease)
* 1.0000894 for every additional dollar in ‘Additional\_charges’ (~0.0089% increase)
* 0.1687 for patients with a complication risk of ‘Low’ (~83.1% decrease)
* 0.7130 for patients with a complication risk of ‘Medium’ (~28.7% decrease)
* 10.6963 for patients with an emergency admission (~970% increase)
* 2.1430 for patients with an observation admission (~114% increase)
* 0.2811 for patients with arthritis (~71.9% decrease)
* 0.3820 for patients with anxiety (~61.8% decrease)
* 0.2692 for patients with asthma (~73.1% decrease)
* 4.5078 for patients who have had a stroke (~351% increase)
* 14.315 for patients who initially sought an MRI (~1330% increase)
* 4.7228 for patients who initially sought a CT scan (~372% increase)
* Constant term of -78.4120 gives an initial odds ratio of ~8.83\*10-35 before its altered by the above multiplicative factors

This model is statistically significant given its high accuracy of 0.9863, LLR p-value of 0.000, and all coefficients in the model having sufficiently low p-values.

For comparison, 36.7% of patients in this data set needed to be readmitted, so the accuracy of 0.9863 demonstrates genuine predictive capabilities that aren’t the product of favoring the majority class.

For practical purposes, this model could be very useful to hospital staff in determining the likelihood a patient will need to be readmitted. While each patient’s health is complex and distinct, knowing in advance they will likely be readmitted within 30 days could alert physicians to more thoroughly examine the patient and take additional actions to reduce the probability of the patient being readmitted. That said, there may be a cluster of patients where readmission is almost inevitable (such as advanced cancers). The hospital would like to serve patients as best they can the first time around, so an advance warning could improve efficiency and patient health.

Unfortunately, this data set is limited in understanding what causes a patient to be readmitted. The vast majority of readmitted patients had initial stays over 30 days, but there’s no data pertaining to what condition(s) kept them for so long. There are higher percentages of readmitted patients among those who had a stroke or admitted primarily for a CT scan or MRI, but nothing beyond that. The strong correlation between length of initial stay and readmitted patients while the other variables have a noticeably smaller impact would benefit from additional data and analysis. That type of imbalance indicates unidentified clustering, likely from serious long term illnesses.

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 whether a patient will need to be readmitted within 30 days and how that data can predict which patients will need readmission, a statistically significant logistic regression model was found to have a very high accuracy of 0.9863 (see the equation in section F.1.).

Hospital staff should use the predictions from this model as an early warning that a patient will likely be readmitted within 30 days. While their readmission may be inevitable in some cases, there are other times where the warning may be critically important for a physician to take extra precautions in ensuring the patient won’t need to return for the same condition within a few weeks. However, the predictive ability shouldn’t be mistaken for causation or absolute, and it will likely deteriorate over time.

The model could be made more resilient to 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, patient diseases, and expected prognosis day-to-day would be very helpful in producing better approximations that are relatively future proof (and less dependent on the length of initial stay).

**G. Panopto Demonstration**

See the attached link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=665a22f7-215f-44ca-941a-b15b00205729

**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.** Bobbitt, Zach. 2020. Statology “The 6 Assumptions of Logistic Regression (With Examples)”. Retrieved April 22, 2024, from <https://www.statology.org/assumptions-of-logistic-regression/>.