**Michael Segaline**

**Dr. Sewell**

**D208: Predictive Modeling**

**12/19/2021**

**Abstract**

Bottom Line Up Front: The baseline odds of a patient not readmitting are 29%.

The purpose of this data analysis is to reduce the number of hospital readmissions for a major hospital chain. The Hospital has provided the dataset of 51 features and 10,000 rows of patient data. The objective is to produce an accurate Logistic Regression model to help hospital staff to better predict which patients will readmit and reduce the number of readmissions. Reducing readmissions will result in money saved not paying fines from outside agencies. Some goals are to clean the data, explore the intended features, get the accuracy of the initial Logistic Regression model, reduce the dimensionality of features, and create a final Logistic Regression model. The metrics of the first model will be compared against the second. Resulting is a Logistic Regression model that can predict patient hospital readmission with 63% accuracy. However, the final model more metrics that could be improved on before it could be a useful model and more data is necessary for certainty.

Logistic Regression is used to describe data and explain relationship between one dependent binary variable and one categorical or continuous variables. Moreover, Logistic regression is a classification algorithm for machine learning and assumes a Bernoulli (binomial) distribution of the independence (RealPython.com). Furthermore, the variables to be considered must not have high correlation between them. However, if the dependent variable is ordinal then Logistic regression requires observation independence. The Logistic Regression equation consists of two parts, the Logit and the Probability (between ‘0’ and ‘1’). The Logit contains the linear representation individual variable coefficients; When nested in the Probability function, outputs a logarithm of odds of achieving ‘0’ or ‘1’ (RealPython.com). Unlike Linear Regression, which is centered on a straight line of best fit, a Logistic Regression generates a sigmoid ‘S’ shaped line (StatQuest 2018). Considering that hospital readmission is a binary feature, Logistic Regression is ideal for prediction over Linear Regression.

One of the benefits of Python is the Pandas library that allows users to import and mine though large datasets with high speed. Another nice aspect is the various statistical libraries and visualization tools. From start to finish, Python can efficiently perform all steps of the Extract Load Transform (ETL) process. Python is an opensource and benefits from the unhindered contributions to the data community. Once fetched and cleaned, the data can be processed by powerful Machine Learning models, by importing the various libraries offered in Python. At the same time, Python offers visualization libraries to create human readable graphs of the data.

The end state, of data preparation, is to produce a standardize dataset containing only the variables of interest, free of missing values and redundant features. Additionally, all categorical features will be converted to binary. From that point, the cleaned dataset can be explored and correctly processed in the initial Logistic Regression model. The selected variables consist of preexisting conditions and possibly trivial patient background information. One of the hopes of this analysis is to discover which features lead to the readmission of patients and other insights of interest to the hospital staff. The variables considered to predict readmission are number of children per patient Children, marital status Marital, population density where the patient resides Population. Alongside features of pre-existing conditions, such as: high blood pressure HighBlood, suffered a Stroke, Anxiety, Diabetes or Asthma. The Professor instructed not to use all the variables in the initial model. The variables selected in the previous MLR performance evaluation, will not be considered because they have been explored and high multi-collinearity exists between them. One of the hopes of this analysis, is to discover which features lead to the readmission of patients and other possible insights.

To achieve this end state, the dataset will first be imported into the Python environment. Second, all missing values will be dropped so they do not impact the model or the summary statistics; This is accomplished via the dropna( ) function. Third, unnecessary features will be dropped from the data set via the drop() function. Fourth, categorical variables will be converted into binary representation via the get\_dummies() function; get\_dummies() creates two columns for every category and drops the first column. To reduce redundancy, the following categorical columns were dropped after the binary conversion “Marital\_Never Married”, “Marital\_Separated”, “Marital\_Widowed”; These features simplify to not married. Additionally, bivariate analysis will be more simplified. Lastly, the data set will be scaled and free of observations exceeding 3 standard deviations. Below is the code used to clean the dataset:

#-- Importing possible libraries --#

import statistics

import numpy as np

import seaborn as sns

from scipy import stats

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib as mpl

from sklearn import linear\_model

import statsmodels.api as sm

%matplotlib inline

import os

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import r2\_score, explained\_variance\_score, confusion\_matrix, accuracy\_score, classification\_report, log\_loss

from math import sqrt

from IPython.display import display

#-- Importing the dataset --#

med\_df\_raw = pd.read\_csv("/Users/lindasegalini/Desktop/WGU/New Program/D208 Predictive Modeling/medical\_clean.csv")

#-- Dropping missing values --#

med\_df\_raw.dropna()

#-- Dropping unessesary features --#

med\_df\_raw = med\_df\_raw.drop(columns = ['Gender','Complication\_risk','Initial\_admin','Additional\_charges','TotalCharge','Initial\_days','VitD\_levels','Age','CaseOrder','Allergic\_rhinitis','Reflux\_esophagitis','Doc\_visits','Full\_meals\_eaten','vitD\_supp','Soft\_drink','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng', 'Area','TimeZone','Job','Income','Services','Item1','Item2','Item3','Item4','Item5','Item6','Item7','Item8', 'Arthritis','Hyperlipidemia','BackPain'])

#-- Changing categorical to binary with get\_dummies() and dropping the first column-#

med\_df = pd.get\_dummies(med\_df\_raw, drop\_first =True)

#-- Dropping the previously combined columns to keep the number of columns down and create a tidier dataset --#

med\_clean = med\_df.drop(columns = ['Marital\_Never Married','Marital\_Separated','Marital\_Widowed'])

#-- Filtering the data frame to remove the values exceeding 3 standard deviations --#

med\_remove\_df = med\_clean[(np.abs(stats.zscore(med\_df)) < 3).all(axis=1)]

#-- Displaying what rows were removed --#

med\_clean.index.difference(med\_remove\_df.index)

#-- Saving a copy of the cleaned dataset --#

med\_clean.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D208 Predictive Modeling/Processed\_med\_data.csv')

#-- Exploring the dataset --#

#-- Examining summary statistics --#

med\_clean.describe()

Now, conducting Univariate and Bivariate analysis between each feature and the Dependent Variable (DV). Displayed in the following graphs are the statistical distribution of each variable. Showcasing the amount of correlation between the DV and each selected feature; The correlation is minuscule, ranging from -1.7% to 2.4%. With that small of a deviation, the frequency differences will not be graphically noticeable. According to the chart, the feature with the highest correlation was ‘Children’ at 2.4%. Interestingly, ‘Diabetes’, ‘Overweight’, and ‘Asthma’ are negatively correlated. The following Correlation Heatmap, looks like a Scuba diving flag, indicating no existence of serious multicollinearity between variables.

Table

Description automatically generated with low confidence

Graphical user interface

Description automatically generated

Below is the distribution of the population variable’s maximum population sample at 122,814. Plus, 75% of the hospital patients came from populations around 14,000 people. When compared to the target variable, population baseline for readmission looks to be slightly higher for higher population areas.

Chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

**The average number of children per patient is approximately two and the baseline for readmission is higher for patients who have at least two children.**

Chart, histogram

Description automatically generatedChart, box and whisker chart

Description automatically generated

**Only 20% of the patients in the data set were married and proportionally represented when compared with the dependent variable. According to the summary statistics, 80% of the patients are not married.**

Chart

Description automatically generatedChart

Description automatically generated

**Approximately 40% of the patients who had high blood pressure; When transposed against the DV, the frequency is about equal.**

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

**Approximately 20% of patients suffered a stroke. The frequency of readmissions associated with stroke as the initial diagnosis is small.**

Chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

**Approximately 71% of patients were overweight and when compared to the dependent variable looks very similar. To interrogate this feature further, it has a slightly negative correlation with the dependent variable at -0.0086.**

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

**Of the patients in the data set, 27.3% had diabetes and is correlated with the target variable at -.003%.**

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

**32.1% of hospital patients had anxiety; Anxiety had a meager 2.1% positive correlation with the target variable.**

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

**Approximately 29% of the patients had asthma. Also, asthma had a -1.7% correlation to the dependent variable.**

Before creating the model, the cleaned dataset will be separated into two data frames, one for testing ‘X’ and the other for training ‘y’. This accomplished by calling the test\_train\_split() function which creates a tuple of four variables when passing ‘X’ and ‘y’ as arguments. This tuple creates the variables: ‘X\_train’,’ X\_test’, ‘y\_train’, ‘y\_test’; In which 20% of the data was allocated to testing and 80% to training. Next, the LogisticRegression() function from the sklearn.linear\_model library was called and the training data was passed as arguments. Next, predictions where forecasted via the predict() function when appended to logistic regression variable. From that point, the sklearn library offers metrics to measure the accuracy of the model, to include the confusion matrix for scoring. The following is the code for the first model’s creation and evaluation:

#-- Split the data into ‘X’ & ‘y’ dataframes --#

X = med\_clean.drop('ReAdmis\_Yes', axis = 1).values

y = med\_clean['ReAdmis\_Yes']

y = y.astype(int)

#-- Make training and testing sets --#

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20,

shuffle=True, random\_state=2)

#-- Initialize the logistic regression model --#

from sklearn.linear\_model import LogisticRegression

log\_reg = LogisticRegression(random\_state=10, solver = 'lbfgs')

log\_reg.fit(X\_train, y\_train)

#-- Make predictions --#

y\_pred = log\_reg.predict(X\_test) # Predictions

y\_true = y\_test # True values

y\_test

#-- Measure accuracy --#

from sklearn.metrics import accuracy\_score

import numpy as np

print("Train accuracy:", np.round(accuracy\_score(y\_train,

log\_reg.predict(X\_train)), 2))

print("Test accuracy:", np.round(accuracy\_score(y\_true, y\_pred), 2))

#-- Make the confusion matrix --#

from sklearn.metrics import confusion\_matrix

cf\_matrix = confusion\_matrix(y\_true, y\_pred)

print("\n First model \nTest confusion\_matrix")

sns.heatmap(cf\_matrix, annot=True, cmap='Blues')

plt.xlabel('Predicted', fontsize=12)

plt.ylabel('True', fontsize=12)

#-- Calculating False Positives (FP), False Negatives (FN), True Positives (TP), andTrue Negatives (TN)

FP = cf\_matrix.sum(axis=0) - np.diag(cf\_matrix)

FN = cf\_matrix.sum(axis=1) - np.diag(cf\_matrix)

TP = np.diag(cf\_matrix)

TN = cf\_matrix.sum() - (FP + FN + TP)

#-- Sensitivity, hit rate, recall, or true positive rate --#

TPR = TP / (TP + FN)

print("The True Positive Rate is:", TPR)

#-- Precision or positive predictive value --#

PPV = TP / (TP + FP)

print("The Precision is:", PPV)

#-- False positive rate or False alarm rate --#

FPR = FP / (FP + TN)

print("The False positive rate is:", FPR)

#-- False negative rate or Miss Rate --#

FNR = FN / (FN + TP)

print("The False Negative Rate is: ", FNR)

#-- Total averages --#

print("")

print("The average TPR is:", TPR.sum()/2)

print("The average Precision is:", PPV.sum()/2)

print("The average False positive rate is:", FPR.sum()/2)

print("The average False Negative Rate is:", FNR.sum()/2)

#-- Probability estimates --#

pred\_proba = log\_reg.predict\_proba(X\_train)

#-- Running Log loss on training --#

print("The Log Loss on Training is: ", log\_loss(y\_train, pred\_proba))

#-- Running Log loss on testing --#

pred\_proba\_t = log\_reg.predict\_proba(X\_test)

print("The Log Loss on Testing Dataset is: ", log\_loss(y\_test, pred\_proba\_t))

Principal Component Analysis or PCA will be used in feature reduction. PCA is non-parametric so it can handle the distributions of the features in the dataset. It is an unsupervised machine learning technique for pattern recognition (towardsdatascience.com). PCA can take a dataset with multiple variables and through a process of orthogonal linear transformation, represent the original variable into a new set of uncorrelated variables called Principal Components (PC), which will retain most of the variation (Pramoditha 2018). Essentially, if an array of numbers is plotted a line of best fit can be drawn through the range of the residuals. This line is called the PC. Then if the same array (or matrix) is multiplied by a scalar, the resulting points will create a line of best fit, in a different direction (Starmer 2018).

Each PC has is its own dimension that can be graphed out and dimensions that are un-important can be excluded. Graphically, the longer the line of the PC, the more variation of datapoints between along the line of best fit (Reneshebred.com 2021). The measure of variance contained in the PC is represented in a metric called an eigenvalue. According to Renshebred.com, if the eigenvalue is one or greater, then it means that it represents all the variation in the dataset (2021). The longer the PC lines, the longer the ‘x’ and ‘y’ lines. Therefore, a bigger scope to view more instances of residuals, leading to more accurate models. PCs with eigenvalues greater than one will be selected for the final model. The scaled and transformed PCA dataset will be fitted to the number of qualifying PCs. That new PCA dataset will be split into testing and training, then used to train a final model.

#--Inspecting eigenvalues on the clean data set --#

med\_clean = StandardScaler().fit\_transform(med\_clean)

pd.DataFrame(med\_clean).head(10)

#--Proportion of Variance (from PC1 to PC9)--#

pca\_out = PCA().fit(med\_clean)

print('The Eigenvalue of Population is:',pca\_out.explained\_variance\_[0] )

print('The Eigenvalue of Children is:',pca\_out.explained\_variance\_[1] )

print('The Eigenvalue of Marital\_Married is:',pca\_out.explained\_variance\_[2] )

print('The Eigenvalue of ReAdmis\_Yes is:',pca\_out.explained\_variance\_[3] )

print('The Eigenvalue of HighBlood\_Yes is:',pca\_out.explained\_variance\_[4] )

print('The Eigenvalue of Stroke\_Yes is:',pca\_out.explained\_variance\_[5] )

print('The Eigenvalue of Overweight\_Yes is:',pca\_out.explained\_variance\_[6] )

print('The Eigenvalue of Diabetes\_Yes is:',pca\_out.explained\_variance\_[7] )

print('The Eigenvalue of Anxiety\_Yes is:',pca\_out.explained\_variance\_[8] )

print('The Eigenvalue of Asthma\_Yes is:',pca\_out.explained\_variance\_[9] )

#-- Output--#

Text

Description automatically generated

#-- ‘Population’, ‘Children’, and ‘Marrital\_Married’ have Eigenvalues greater than one --#

#-- Conducting feature scaling on the 'X' and 'y' dataframes --#

sc = StandardScaler()

X\_scaled = sc.fit\_transform(X)

#-- Applying PCA with three Principal components --#

from sklearn.decomposition import PCA

pca = PCA(n\_components=3)

X\_pca = pca.fit\_transform(X\_scaled)

#-- Creating the final model --#

#-- Spliting into training and testing --#

from sklearn.model\_selection import train\_test\_split

X\_train\_pca, X\_test\_pca, y\_train, y\_test = train\_test\_split(X\_pca, y, test\_size=0.20,

shuffle=True, random\_state=2)

#-- Initialize the logistic regression model --#

from sklearn.linear\_model import LogisticRegression

log\_reg2 = LogisticRegression(random\_state=10, solver = 'lbfgs')

#-- Train the model --#

log\_reg2.fit(X\_train\_pca, y\_train)

#-- Make predictions --#

y\_pred = log\_reg2.predict(X\_test\_pca) # Predictions

y\_true = y\_test # True values

#-- Measure accuracy --#

from sklearn.metrics import accuracy\_score

print("Train accuracy:", np.round(accuracy\_score(y\_train,

log\_reg2.predict(X\_train\_pca)), 2))

print("Test accuracy:", np.round(accuracy\_score(y\_true, y\_pred), 2))

#-- Make the confusion matrix --#

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_true, y\_pred)

print("\nTest confusion\_matrix")

sns.heatmap(cf\_matrix, annot=True, cmap='Blues')

plt.xlabel('Predicted', fontsize=12)

plt.ylabel('True', fontsize=12)

#-- Generating the data for the Logistic Regression Equation --#

print ('The intercept and coefficents the logistic regression model are: ',log\_reg2.intercept\_, ' = ', log\_reg2.coef\_ )

#--Calculating False Positives (FP), False Negatives (FN), True Positives (TP) & True Negatives (TN)

FP = cm.sum(axis=0) - np.diag(cm)

FN = cm.sum(axis=1) - np.diag(cm)

TP = np.diag(cm)

TN = cm.sum() - (FP + FN + TP)

#-- Sensitivity, hit rate, recall, or true positive rate --#

TPR = TP / (TP + FN)

print("The True Positive Rate is:", TPR)

#-- Precision or positive predictive value --#

PPV = TP / (TP + FP)

print("The Precision is:", PPV)

#-- False positive rate or False alarm rate --#

FPR = FP / (FP + TN)

print("The False positive rate is:", FPR)

#-- False negative rate or Miss Rate --#

FNR = FN / (FN + TP)

print("The False Negative Rate is: ", FNR)

#-- Total averages --#

print("")

print("The average TPR is:", TPR.sum()/2)

print("The average Precision is:", PPV.sum()/2)

print("The average False positive rate is:", FPR.sum()/2)

print("The average False Negative Rate is:", FNR.sum()/2)

#-- Make a prediction for every instance in the cleaned dataset--#

pred\_proba2 = log\_reg2.predict\_proba(X\_train\_pca)

#-- Running Log loss on training --#

print("The Log Loss on Training is: ", log\_loss(y\_train, pred\_proba2))

#-- Running Log loss on testing --#

pred\_proba\_t2 = log\_reg.predict\_proba(X\_test)

print("The Log Loss on Testing Dataset is: ", log\_loss(y\_test, pred\_proba\_t2))

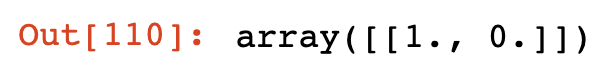
#--Predict hospital readmission if a person from a high population of 100,000 people, who is not married with no children--#

guess1 = [[100000,0,0]]

pred\_1 = log\_reg2.predict\_proba(guess1)

pred\_1

#--Predicted re-admission: Yes 100% probability --#

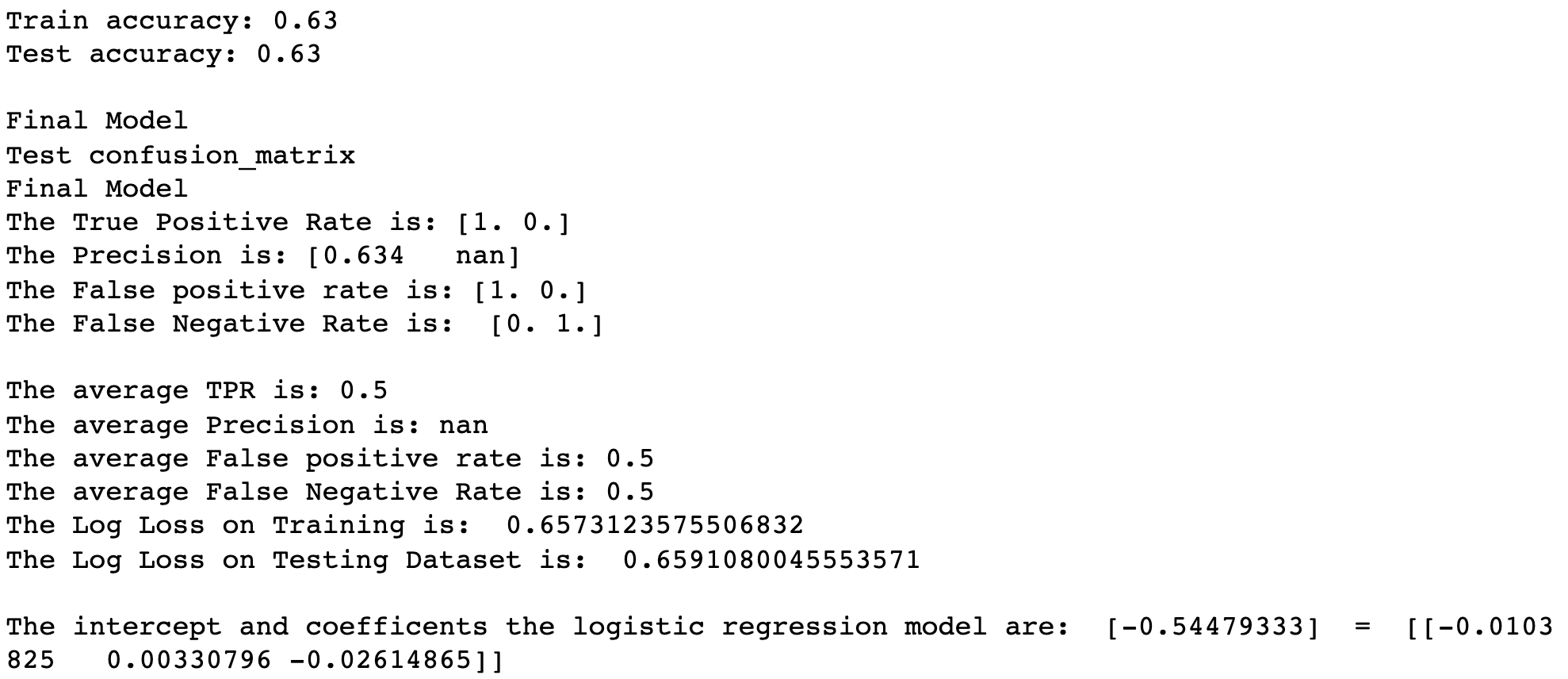


Chart

Description automatically generatedChart

Description automatically generated

Text, letter

Description automatically generated

According to towardsdatascience.com, “The log loss is indicative of how close the prediction probability is to the corresponding true value 0 or 1 in case of binary classification the more predictive probability diverges from actual value the higher the Log Loss”(2019). The closer the score is to one, the better. Noticing that the Precision, is within two percent of the Log Loss score, indicates that the model’s accuracy is near matching on two separate metrics. The log loss is indicative of how close the prediction probability is to the corresponding true value ‘0’ or ‘1’ in case of binary classification the more predictive probability diverges from actual value the higher the log loss.

The logit of the equation is: ƒ(x1,x2) = -0.54 ReAdmis\_Yes – 0.01 Population + 0.003 Children – 0.026 Marrital\_Married.

The probabilities: p(x1,x2) = 1/(1+exp(-ƒ(x1,x2)))

The full equation of the final Logistic Regression model is:

p(x1,x2) = 1/(1+exp(-(-0.54 ReAdmis\_Yes – 0.01 Population + 0.003 Children – 0.026 Marrital\_Married)))

The target variable coefficient sets the baseline likelihood -54.6% against hospital readmission. As the intercept the ß0 = log of odds when all ßn are equal to zero. For every change in one unit of the population, the goes against readmission by 1%. Assuming all variables are equal to zero, for every 1 unit of change in the ‘Population’, effects the DV by 0.001%; For every 1 child of a patient, effects the DV by 0.003%. Lastly, for every 1 unit of change in ‘Married’ effects the DV by 2.6 %. The Odds Ratio can quickly be determined by exponentiating the coefficients of each variable (marinstatlectures 2021). The DV has the largest coefficient of note when expatiated -0.54^(2) = -29%. Therefore, the baseline odds of a random patient in the population not readmitting are 29%. The Odds ratios of the remaining variables are too small to have a meaningful impact on a binary unit change of 1.

In final analysis, the populations density of patients from this dataset is reflective of small towns with possibly less hospital access. Other considerations for population and marriage may be more communal support for patients; that one person who's there to ensure the patient goes to the hospital. Considering, the coefficient is less than one percent for patients with children, it was initially the highest of correlation to the DV (of the final PCs). Aside from 0.001 differences in the Log of Loss, comparing both models side by side, they are almost identical. The True and False Positives Rates are 100%. At 50 / 50, the average True and False Positive Rates, reduce the model to little more than a coin flip for prediction. Say the model did predict correct, there remains a 37% (1 – p) probability to being wrong. Even if the model’s True and False Positive rate is improved, that is not much better than human intuition. One of the limitations is the small dataset and may not be reflective of the true patient population. More time plus data is needed to properly explore other features and may yield a better model.

Work Cited

Bedre, R. (2021, November 7). *Principal Component Analysis (PCA) and visualization using python (detailed guide with example)*. Data science blog. Retrieved December 30, 2021, from https://www.reneshbedre.com/blog/principal-component-analysis.html

Dembla, G. (2021, December 3). *Intuition behind log-loss score*. Medium. Retrieved December 30, 2021, from https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a

joshstarmer. (2018, April 2). *StatQuest: Principal Component Analysis (PCA), step-by-step*. YouTube. Retrieved December 30, 2021, from https://www.youtube.com/watch?v=FgakZw6K1QQ

marinstatlectures, M. (2021, January 12). *5.8 logistic regression: Why do the model coefficients give us odds ratios (or)?* YouTube. Retrieved December 30, 2021, from https://www.youtube.com/watch?v=eqLHHVbf\_do

Pramoditha, R. (2021, September 20). *How do you apply PCA to logistic regression to remove multicollinearity?* Medium. Retrieved December 30, 2021, from https://towardsdatascience.com/how-do-you-apply-pca-to-logistic-regression-to-remove-multicollinearity-10b7f8e89f9b

Real Python. (2020, November 24). *Logistic regression in python*. Real Python. Retrieved December 30, 2021, from https://realpython.com/logistic-regression-python/