Predictive Modeling – D208

Task 2

Western Governor’s University

Performance Assessment

Matthew Morgan

Student ID: 010471280

11/28/2022

**Part I: Research Question**

A1. What variables lead to patient readmissions?

A2. Hospitals are penalized by an external organization for excessive readmissions. To help prevent the hospital from being penalized we need to identify factors that lead to patient readmissions. Once those factors are identified we can hopefully find ways to reduce patient readmissions.

**Part II: Method Justification**

B1. Logistic regression predicts whether something is True or False, instead of something continuous. Therefore, we are using a categorical variable for logistic regression. Logistic regression will fit an s-shaped line to the predictions to produce predictions. This line allows us to predict the probability of a prediction and use the likelihood of that prediction to classify it within the categorical variable.

“The logistic regression model is based on different assumptions than linear regression:

* It is based on the Bernoulli distribution because the dependent variable is binary.
* The predict values are restricted to a range of nominal values like ‘Yes’ and ‘No’, not Small, Medium, Large.
* It predicts the probability of particular outcomes rather than the outcome itself.
* It is the logarithm of the odds of achieving 1.” (Sewell, 2022)

B2. I am using Python as it’s the language I am most comfortable with and its versatility. Python has many packages and libraries available to run logistic regression easily and quickly. Because of its versatility, you can run the whole ETL pipeline in one python script.

B3. Logistic regression is used specifically for categorical variables and can be used in conjunction with other categorical or continuous variables to make predictions. Because we are using a categorical variable as our dependent variable in this task, we want to just logistic regression and build a model that can predict future patient readmissions.

**Part III: Data Preparation**

C1.

* Import medical\_clean.csv into Jupyter Notebook
* Build boxplots to check for outliers
* Convert Yes/No into quantitative data
* Use pd.get\_dummies to convert categorical variables into quantitative data
* Remove redundant columns (Marital\_Divorced, Gender\_Female to reduce possibilities of multicollinearity
* Rename columns from pd.get\_dummies by replacing spaces with underscores
* Run univariate stats script to calculate how many rows, missing values, unique values, data type, Mean, Mode, Min, Median, Max, Standard Deviation, Skew, Kurtosis for each numeric column.
* Export cleaned data set

C2. The target categorical variable I chose for this task was ReAdmis which has been converted to ReAdmis\_numeric. Because it is categorical we can use it for logistic regression. The predictor variables I chose were the following; Initial\_days, vitD\_supp, Children, Income, Full\_meals\_eaten, Additional\_charges, TotalCharge, VitD\_levels, Age, Doc\_visits, HighBlood\_numeric, Stroke\_numeric, Arthritis\_numeric, Diabetes\_numeric, Hyperlipidemia\_numeric, BackPain\_numeric, Allergic\_rhinitis\_numeric, Reflux\_esophagitis\_numeric, Asthma\_numeric, Overweight\_numeric, Anxiety\_numeric, Marital\_Married, Marital\_Never\_Married, Marital\_Separated, Marital\_Widowed, Services\_Blood\_Work, Services\_CT\_Scan, Services\_Intravenous, Services\_MRI, Gender\_Male, Gender\_Nonbinary, Initial\_admin\_Elective\_Admission, Initial\_admin\_Emergency\_Admission, Initial\_admin\_Observation\_Admission, Complication\_risk\_High, Complication\_risk\_Low, and Complication\_risk\_Medium

I included screenshots of summary statistics below:

|  |
| --- |
| Summary statistics |
|  |

The summary stats show us that the dataset has many continuous variables, due to the mean/max/mode being outside of 0 and 1. Which allowed me to go back and change the Yes/No columns into 0s and 1s, and also use pd.get\_dummies to one-hot encode other categorical columns.

The summary statistics overall show us that our average patient has 2 children (with a standard deviation of 2.16), has an income of $40k/yr (with a standard deviation of $28,521), eats 1 full meal a day while in the hospital (with a standard deviation of 1), receives $12,934 in additional charges (with a standard deviation of $6,542), spends 34 days on their initial stay in the hospital (with a standard deviation of 26 days), receives $5,312 in total charges (with a standard deviation of $2180), has Vitamin D levels of 17.96 ng/mL upon admission (with a standard deviation of 2), is 53 years old (with a standard deviation of 21) , and is visited by their doctor 5 times during their stay (with a standard deviation of 1).

C3. To prepare the data for analysis I re-expressed some categorical variables on my own, and used pd.get\_dummies to automate one-hot encoding of others. Code snippets are below.

|  |
| --- |
| Re-expression of categorical variables |
| #Data Wrangling; turn categorical values into quantitative data  df['ReAdmis\_numeric'] = df['ReAdmis']  dict\_ReAdmis = {"ReAdmis\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_ReAdmis, inplace=True)  df['Soft\_drink\_numeric'] = df['Soft\_drink']  dict\_Soft\_drink = {"Soft\_drink\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_Soft\_drink, inplace=True)  df['HighBlood\_numeric'] = df['HighBlood']  dict\_HighBlood = {"HighBlood\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_HighBlood, inplace=True)  df['Stroke\_numeric'] = df['Stroke']  dict\_stroke = {"Stroke\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_stroke, inplace=True)  df['Arthritis\_numeric'] = df['Arthritis']  dict\_arthritis = {"Arthritis\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_arthritis, inplace=True)  df['Diabetes\_numeric'] = df['Diabetes']  dict\_diabetes = {"Diabetes\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_diabetes, inplace=True)  df['Hyperlipidemia\_numeric'] = df['Hyperlipidemia']  dict\_hyperlipidemia = {"Hyperlipidemia\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_hyperlipidemia, inplace=True)  df['BackPain\_numeric'] = df['BackPain']  dict\_backpain = {"BackPain\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_backpain, inplace=True)  df['Allergic\_rhinitis\_numeric'] = df['Allergic\_rhinitis']  dict\_allergies = {"Allergic\_rhinitis\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_allergies, inplace=True)  df['Reflux\_esophagitis\_numeric'] = df['Reflux\_esophagitis']  dict\_reflux = {"Reflux\_esophagitis\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_reflux, inplace=True)  df['Asthma\_numeric'] = df['Asthma']  dict\_asthma = {"Asthma\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_asthma, inplace=True)  df['Overweight\_numeric'] = df['Overweight']  dict\_Overweight = {"Overweight\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_Overweight, inplace=True)  df['Anxiety\_numeric'] = df['Anxiety']  dict\_Anxiety = {"Anxiety\_numeric": {"No": 0, "Yes": 1}}  df.replace(dict\_Anxiety, inplace=True) |

|  |
| --- |
| Pd.get\_dummies one-hot encoding |
| df = pd.get\_dummies(df, columns=["Marital", "Services", "Gender", "Initial\_admin", "Complication\_risk"]) |

I used both methods for the practice and to get experience with both.

C4. Univariate and Bivariate visualizations are below:

|  |
| --- |
| Univariate Visualizations |
|  |

|  |
| --- |
| Bivariate Visualizations |
|  |

C5. Prepared data set uploaded as part of submission.



**Part IV: Model Comparison and Analysis**

D1. Initial regression model with variables identified in C2

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

D2. The log-likelihood value is -329.14 and the Pseudo R-Squared is 0.9499. These numbers will be used later to compare them to the reduced model. The LLR p-value is 0.000 which tells us that the model is “useful” in predicting the values of the response variable.

To reduce the model I am going to run a VIF and create a new model with a reduced amount of variables. The VIF will help me determine the multicollinearity of the variables used in the initial model. I can also calculate the AIC of the models and that can help me to determine what variables I want to continue with in building a model to predict patient readmissions.

Results of VIF from initial model below:

Table

Description automatically generated with medium confidence

D3. Results of reduced logistic regression model

Table

Description automatically generated

**E.  Analyze the data set.**

E1. For the initial model I chose to use all the variables that related to patient demographic and health condition. In the reduced model I removed variables that had high p-values and were redundant based on have multicollinearity with other variables whether identified through running the VIF or looking at our bivariate visualizations. This biggest example of redundant data that could be removed just from looking at the visualizations is TotalCharges because that data is essentially based on Initial\_days already.

When comparing our initial model and reduced model we want to use a few key stats from our logit summary.

The initial model gives us a log-likelihood value of -329.14 and Pseudo R-Squared value of 0.9499.

The reduced model gives us a log-likelihood value of -457.04 and Pseudo R-Squared value of 0.9305.

Both models have a LLR p-value of 0.000.

Based on comparing the log-likelihood and Pseudo R-Squared values, our initial model is a better fit for making predictions. This is also backed up by the AIC scores and confusion matrix from each model which are included below. The initial model scores as a “better” model according to both AIC and the accuracy calculation from the confusion matrix.

E2.

|  |  |
| --- | --- |
| Initial Model | Reduced Model |
| AIC Score and Code:  #Calculating AIC of Initial Model  from sklearn.linear\_model import LinearRegression  import statsmodels.api as sm  #define response variable  y = df['ReAdmis\_numeric']  #define predictor variables  x = df[['Initial\_days', 'vitD\_supp', 'Children', 'Income', 'Full\_meals\_eaten', 'Additional\_charges', 'TotalCharge', 'VitD\_levels', 'Age', 'Doc\_visits', 'HighBlood\_numeric', 'Stroke\_numeric', 'Arthritis\_numeric', 'Diabetes\_numeric', 'Hyperlipidemia\_numeric', 'BackPain\_numeric', 'Allergic\_rhinitis\_numeric', 'Reflux\_esophagitis\_numeric', 'Asthma\_numeric', 'Marital\_Married', 'Marital\_Never\_Married', 'Marital\_Separated', 'Marital\_Widowed', 'Services\_Blood\_Work', 'Services\_CT\_Scan', 'Services\_Intravenous', 'Services\_MRI', 'Gender\_Male', 'Gender\_Nonbinary', 'Initial\_admin\_Elective\_Admission', 'Initial\_admin\_Emergency\_Admission', 'Initial\_admin\_Observation\_Admission', 'Complication\_risk\_High', 'Complication\_risk\_Low', 'Complication\_risk\_Medium']]  #add constant to predictor variables  x = sm.add\_constant(x)  #fit regression model  model = sm.OLS(y, x).fit()  #view AIC of model  print(model.aic)  **Score:** 897.5899676679401  (Statology, 2021) | AIC Score and Code:  #Calculating AIC of Reduced Model #1  #define response variable  y = df['ReAdmis\_numeric']  #define predictor variables  x = df[['Initial\_days', 'Children', 'Stroke\_numeric', 'Asthma\_numeric', 'Overweight\_numeric']]  #add constant to predictor variables  x = sm.add\_constant(x)  #fit regression model  model = sm.OLS(y, x).fit()  #view AIC of model  print(model.aic)  **Score:** 918.392404996368  (Statology, 2021) |
| Confusion Matrix and Code:  #Confusion Matrix for Initial Model  conf\_matrix = mdl\_readmis\_vs\_variables.pred\_table()  print(conf\_matrix)  from statsmodels.graphics.mosaicplot import mosaic  mosaic(conf\_matrix)  #Calculating accuracy: the proportion of correct predictions  TN = conf\_matrix[0,0]  TP = conf\_matrix[1,1]  FN = conf\_matrix[1,0]  FP = conf\_matrix[0,1]  acc = (TN + TP) / (TN + TP + FN + FP)  print('Accuracy:', acc)  #Sensitivity: proportion of true positives  sens = TP / (FN + TP)  print('Sensitivity:', sens)  #Specificity: proportion of true negatives  spec = TN / (TN + FP)  print('Specificity:', spec) | Confusion Matrix and Code:  #Confusion Matrix for Reduced Model  conf\_matrix = mdl\_readmis\_vs\_variables1.pred\_table()  print(conf\_matrix)  from statsmodels.graphics.mosaicplot import mosaic  mosaic(conf\_matrix)  #Calculating accuracy: the proportion of correct predictions  TN = conf\_matrix[0,0]  TP = conf\_matrix[1,1]  FN = conf\_matrix[1,0]  FP = conf\_matrix[0,1]  acc = (TN + TP) / (TN + TP + FN + FP)  print('Accuracy:', acc)  #Sensitivity: proportion of true positives  sens = TP / (FN + TP)  print('Sensitivity:', sens)  #Specificity: proportion of true negatives  spec = TN / (TN + FP)  print('Specificity:', spec) |

Reduced Model Residual Standard Error (Techhelpnotes, 2022)

Graphical user interface, text, application, Word

Description automatically generated

Reduced Model Residual Plot

Graphical user interface, application

Description automatically generated

E3.

|  |
| --- |
| Initial Model |
| #Initial Logistic Regression Model  mdl\_readmis\_vs\_variables = logit("ReAdmis\_numeric ~ Initial\_days + vitD\_supp + Children + Income + Full\_meals\_eaten + Additional\_charges + TotalCharge + VitD\_levels + Age + Doc\_visits + HighBlood\_numeric + Stroke\_numeric + Arthritis\_numeric + Diabetes\_numeric + Hyperlipidemia\_numeric + BackPain\_numeric + Allergic\_rhinitis\_numeric + Reflux\_esophagitis\_numeric + Asthma\_numeric + Overweight\_numeric + Anxiety\_numeric + Marital\_Married + Marital\_Never\_Married + Marital\_Separated + Marital\_Widowed + Services\_Blood\_Work + Services\_CT\_Scan + Services\_Intravenous + Services\_MRI + Gender\_Male + Gender\_Nonbinary + Initial\_admin\_Elective\_Admission + Initial\_admin\_Emergency\_Admission + Initial\_admin\_Observation\_Admission + Complication\_risk\_High + Complication\_risk\_Low + Complication\_risk\_Medium", data=df).fit()  print(mdl\_readmis\_vs\_variables.params)  mdl\_readmis\_vs\_variables.summary() |

|  |
| --- |
| VIF to reduce model |
| #Variable Selection  # Checking for the VIF values of the variables.  from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  X = df[['Initial\_days', 'vitD\_supp', 'Children', 'Income', 'Full\_meals\_eaten', 'Additional\_charges', 'TotalCharge', 'VitD\_levels', 'Age', 'Doc\_visits', 'HighBlood\_numeric', 'Stroke\_numeric', 'Arthritis\_numeric', 'Diabetes\_numeric', 'Hyperlipidemia\_numeric', 'BackPain\_numeric', 'Allergic\_rhinitis\_numeric', 'Reflux\_esophagitis\_numeric', 'Asthma\_numeric', 'Marital\_Married', 'Marital\_Never\_Married', 'Marital\_Separated', 'Marital\_Widowed', 'Services\_Blood\_Work', 'Services\_CT\_Scan', 'Services\_Intravenous', 'Services\_MRI', 'Gender\_Male', 'Gender\_Nonbinary', 'Initial\_admin\_Elective\_Admission', 'Initial\_admin\_Emergency\_Admission', 'Initial\_admin\_Observation\_Admission', 'Complication\_risk\_High', 'Complication\_risk\_Low', 'Complication\_risk\_Medium']]  # VIF dataframe  vif\_data = pd.DataFrame()  vif\_data["feature"] = X.columns    # calculating VIF for each feature  vif\_data["VIF"] = [variance\_inflation\_factor(X.values, i)  for i in range(len(X.columns))]    print(vif\_data) |

(GeeksforGeeks, 2019)

|  |
| --- |
| Reduced Model |
| #Reduced Model removing complication risk, initial admin, services, demographics, and charges columns due to VIF being high, redundancy, and high p-value  mdl\_readmis\_vs\_variables1 = logit("ReAdmis\_numeric ~ Initial\_days + Children + Stroke\_numeric + Asthma\_numeric", data=df).fit()  print(mdl\_readmis\_vs\_variables1.params)  mdl\_readmis\_vs\_variables1.summary() |

**Part V: Data Summary and Implications**

F1. My linear regression equation:

Y = -57.9744 + 1.0651 (Initial\_days) + 0.0698 (Children) + 1.2735 (Stroke\_numeric) + -0.9440 (Asthma\_numeric)

This line means for every 1 unit of:

Initial\_days, ReAdmis\_numeric will increase 1.0651 units

Children ReAdmis\_numeric will increase 0.0698 units

Stroke\_numeric ReAdmis\_numeric will increase 1.2735 units

Asthma\_numeric ReAdmis\_numeric will decrease 0.9440 units

As stated previously, when comparing our initial model and reduced model we want to use a few key stats from our logit summary.

The initial model gives us a log-likelihood value of -329.14 and Pseudo R-Squared value of 0.9499.

The reduced model gives us a log-likelihood value of -457.04 and Pseudo R-Squared value of 0.9305.

Both models have a LLR p-value of 0.000.

Based on comparing the log-likelihood and Pseudo R-Squared values, our initial model is a better fit for making predictions. This is also backed up by the AIC scores and confusion matrix from each model which are included below. The initial model scores as a “better” model according to both AIC and the accuracy calculation from the confusion matrix.

Based on these key stats, I would say that these models are not practically significant, though the initial model could be statistically significant.

Neither model should be used to make predictions about what patients will be readmitted. Both models are very limited in what they can do. I believe this mainly has to do with the dataset. More data and different data needs to be captured.

F2. Based on my results, there really isn’t a course of action to be recommended. The initial model is too robust and complex, and no predictions can really be made. The reduced model is less accurate than the initial model so we wouldn’t want to use that for predictions either. The course of action would be to start back at square 1, reduce our model with a different method, and re-evaluate what data we capture moving forward.

**Part VI: Demonstration**

G.  https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=57dfea46-a1e1-499f-beb9-af5800362f65

H.

Techhelpnotes, 2022 <https://techhelpnotes.com/residual-standard-error-of-a-regression-in-python/>

GeeksforGeeks, 2019 <https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/>

Statology, 2021 <https://www.statology.org/aic-in-python/>

I.  Sewell, William. (2022). *D208 Predictive Modeling Webinar Episode 3* [Slide 17]