**DS 7333; Case Study #2: Imputation and Logistic Regression – Diabetes Hospital readmission prediction.**

Allen Miller, Vijayasrikanth Kaniti, Venkata ‘MG’ Vanga.

September 17, 2021

Professor: Dr. R. D. Slater

# Introduction

The case study focuses on imputation of missing variables and prediction of hospital readmission from the diabetes data set provided. The dataset consists of 101,766 rows of data categorized into 50 variables. Among the 50 variables 13 are numeric and the remaining 37 are categorical variables (See Figure A1 in Appendix). As the prediction of hospital readmission is a classification problem, Logistic regression is considered appropriate and the equation used for classification is shown in Eq. 1 below.

Where,

– is the predicted hospital readmission classification.

– represent the coefficients of different features used in prediction using logistic regression.

– represent the corresponding features used in prediction.

The hospital readmission variable has three categories for patients with no readmission, patients with readmission below 30 days and patients with readmission above 30 days. The study

# Methods

**Missing Values:**

The dataset is checked for missing values and the summary of missing values and the percentage of missing values is shown in Table A1. The variables (columns) with more than 30% of missing values are removed from the dataset and the remaining missing values are imputed as shown in Table A1; Variables ‘diag\_1’, ‘diag\_2’ and ‘diag\_3’ are removed from the dataset as the data in these columns does not correspond to any particular type of data (*The authors believe the data has been deliberately tarnished for imputation study purpose.*). Variable ‘race’ has been imputed with mode value of the column (‘Caucasian’). The final dataset that is used for logistic regression has 44 variables.

**One Hot Encoding:**

The 44 variables after the imputation for missing values are further evaluated for one hot encoding and variables which just gather the data for the patient ['encounter\_id' and 'patient\_nbr'] has been dropped from the dataset as these variables only describe the patient details and would not help in the classification of readmission. The patients readmitted may have same patient number and the removal of the ‘patient\_nbr’ variable makes the patients independent from each other. Further analysis of the data set the discharge\_disposition\_ids [11, 13, 14, 19, 20 and 21] are related to death or hospice, the rows with these discharge\_disposition\_id rows are removed from the dataset. This reduced the dataset to 99,343 rows [2,423 deleted rows].

Pandas ‘get\_dummies’ option is used to convert all the data variables used for prediction to one hot encoded data (See Figure A2). A test/train split of 65%/35% is selected along with stratified 5-fold cross validation to run the logistic regression model. An inverse regularization strength   
C = 0.0278 is selected from initial investigation of the training data cross validation accuracy score.

The log loss function for the logistic regression is shown in shown in Eq.2

Where if y = 1 then only predictions close to 1 contribute significantly to the loss and vise versa.

# Results

The readmission prediction with the methods described above is shown Figure 1. An overall accuracy of 57% is obtained for the test data. The precession for the logistic regression model is in the range of 45-59% but the recall for readmission category less than 30 days is only 2%, this can be attributed to the unbalance in the data set where the <30 days readmission category is only 11% of the entire data. The confusion matrix heat map is presented in Figure 2. The feature that are important for the readmission prediction is shown in Figure 3. Admission Source, Discharge disposition Id, admission type id and diabetes med are some of important features for classification of readmission.

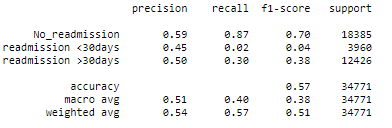


Figure 1: Logistic Regression Classification for Hospital Readmission

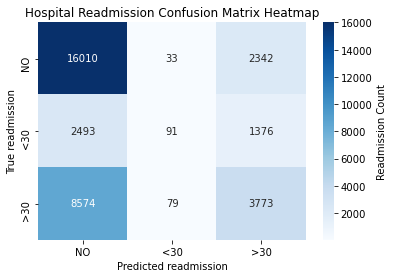


Figure 2: Hospital Readmission Confusion Matrix.

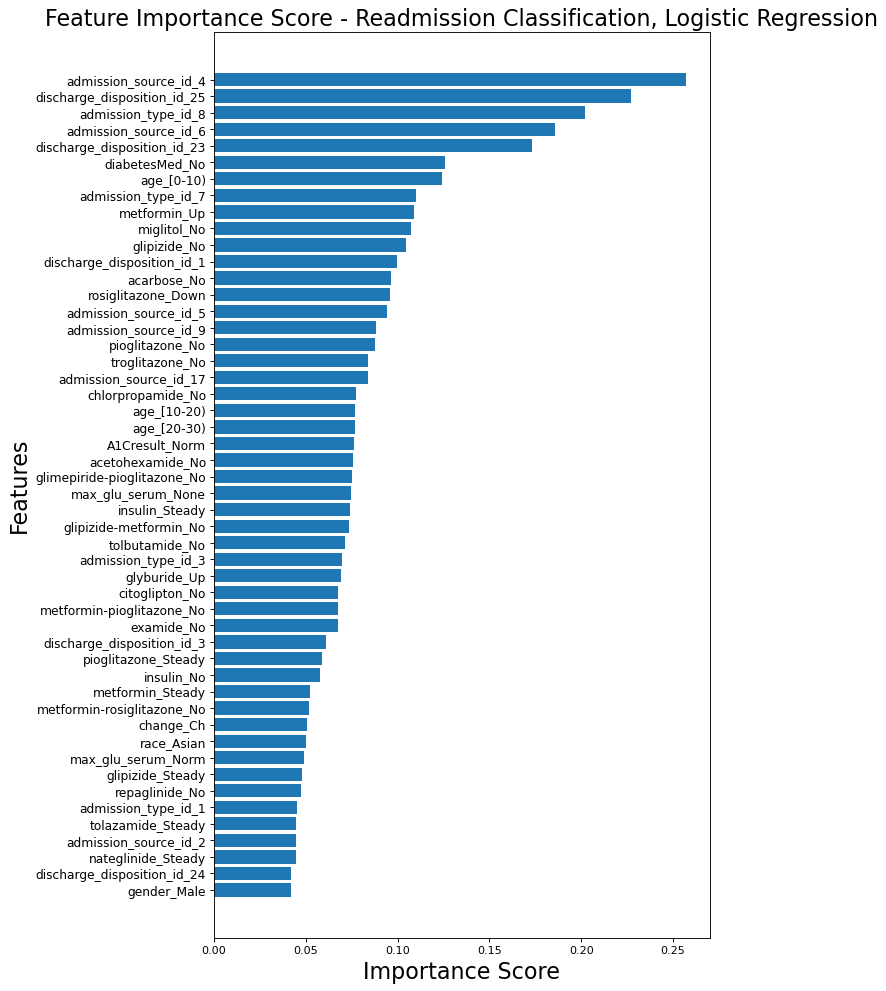


Figure : Hospital Readmission Feature Importance.

# Conclusion

Based on the evaluation of results it can be concluded that ‘admission source’, ‘discharge disposition Id’, ‘admission type id’ and ‘diabetes med’ are some of the important features in classification of the hospital readmission. Further analysis may be useful with the assistance of a domain subject matter experts. This would allow an in-depth look at the levels of discharge disposition id and admission type id to determine which levels were valid and/or duplicates with different names.

# Appendix:

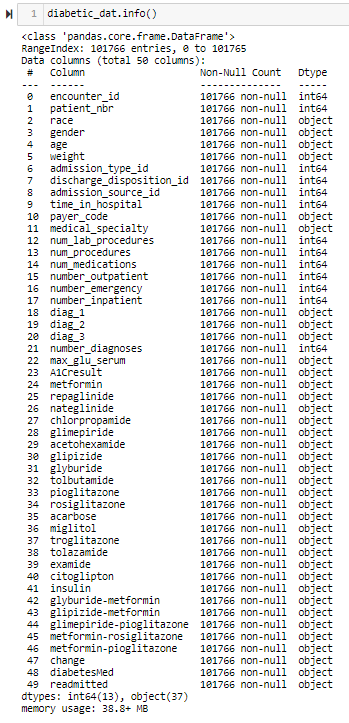
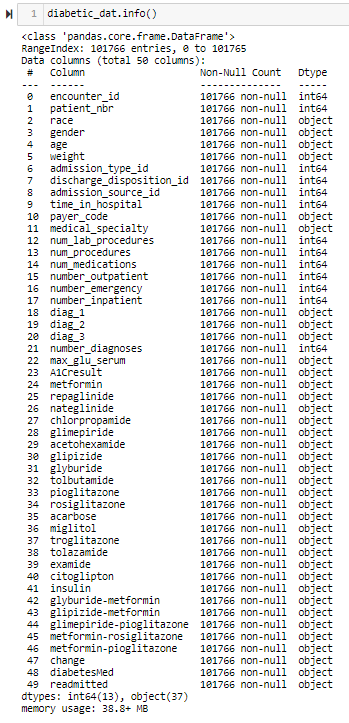


Figure A1: Diabetes dataset variable information.

Table A: Missing variable count and percentages.

| **Missing Variable** | **Missing Count** | **Missing Count (%)** | **Imputation method** |
| --- | --- | --- | --- |
| **diag\_1** | **21** | **0.02%** | **Variable removed** |
| **diag\_2** | **358** | **0.35%** | **Variable removed** |
| **diag\_3** | **1423** | **1.40%** | **Variable removed** |
| **race** | **2273** | **2.23%** | **Mode** |
| **payer\_code** | **40256** | **39.56%** | **Variable removed** |
| **medical\_specialty** | **49949** | **49.08%** | **Variable removed** |
| **weight** | **98569** | **96.86%** | **Variable removed** |



Figure A2: Diabetes dataset variable information after imputation and one hot encoding.

# Code

# Diabetic Dataset - Hospital readmission prediction (Case Study #2: DS7331)

SMU Student collaboration:

- Allen Miller

- Vijay Kaniti

- Venkata 'MG' Vanga

### Links

- https://github.com/ResidentMario/missingno

```python

import pandas as pd

import numpy as np

from copy import deepcopy

import missingno as msno

import seaborn as sns

#reading in the CSV files

diabetic\_dat = pd.read\_csv('https://raw.githubusercontent.com/VenkataVanga/Quantifying-the-World-DS7333-/'\

'main/CS%232\_Diabetes/diabetic\_data.csv')

#diabetic\_dat

```

```python

diabetic\_dat.info()

```

```python

diabetic\_na = deepcopy(diabetic\_dat)

diabetic\_na = diabetic\_na.replace('?', np.nan)

```

```python

missing = diabetic\_na.isna().sum()

missing = pd.DataFrame(missing, columns = ['missing'])

missing = missing[(missing.T !=0).any()]

missing = missing.sort\_values('missing')

missing

```

```python

import matplotlib.pyplot as plt

from matplotlib.pyplot import figure

missing.plot(kind = 'bar', legend = False)

plt.ylabel('Missing Value Count', fontsize=18)

plt.xlabel('Variables', fontsize=18)

plt.xticks(rotation='90')

for index,data in enumerate(missing['missing']):

plt.text(x=index , y =data+2 , s=f"{data}" , fontdict=dict(fontsize=15), ha='center')

plt.figure(figsize=(10, 25))

plt.show

```

```python

#msno.heatmap(diabetic\_na)

```

```python

diabetic\_na.isnull().sum()/len(diabetic\_na)\*100

```

### Imputation

```python

# removing data columns where more than 30% of the data is missing.

diabetic\_na = diabetic\_na.drop(['weight','payer\_code','medical\_specialty'], axis = 1)

```

```python

diabetic\_na.info()

```

```python

#check to see how many values of na can be dropped.

#diabetic\_na.dropna()

```

```python

# filling 'race' with Caucasian as 75% of 'race' is caucasian.

diabetic\_na['race'].fillna('Caucasian')

```

```python

# removing data columns where tarnished data values are added.

diabetic\_na = diabetic\_na.drop(['diag\_1','diag\_2','diag\_3'], axis = 1)

```

```python

diabetic\_na.info()

```

```python

diabetic\_na = diabetic\_na.drop(['encounter\_id','patient\_nbr'], axis=1)

```

```python

#Looking at the IDs\_mapping.csv we can see that 11,13,14,19,20,21 are related to death or hospice.

#These samples are removed from the predictive model.

diabetic\_na = diabetic\_na.loc[~diabetic\_na.discharge\_disposition\_id.isin([11,13,14,19,20,21])]

```

```python

#Changing readmitted variable to 0(No),1(<30),2(>30)

readmitted = {'<30':1,'NO':0,'>30':2}

diabetic\_na.readmitted = [readmitted[item] for item in diabetic\_na.readmitted]

```

```python

# Numerical variables

cols\_num = ['time\_in\_hospital','num\_lab\_procedures', 'num\_procedures', 'num\_medications',

'number\_outpatient', 'number\_emergency', 'number\_inpatient','number\_diagnoses']

```

```python

# Categorical Variables

cols\_cat = ['race', 'gender',

'max\_glu\_serum', 'A1Cresult',

'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',

'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',

'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',

'tolazamide', 'insulin',

'glyburide-metformin', 'glipizide-metformin',

'glimepiride-pioglitazone', 'metformin-rosiglitazone',

'metformin-pioglitazone', 'change', 'diabetesMed', 'payer\_code']

```

```python

# Conversion of numeric variables into strings

cols\_cat\_num = ['admission\_type\_id', 'discharge\_disposition\_id', 'admission\_source\_id']

diabetic\_na[cols\_cat\_num] = diabetic\_na[cols\_cat\_num].astype('str')

```

```python

X = diabetic\_na.drop('readmitted', axis=1).copy()

y = diabetic\_na['readmitted'].copy()

X = pd.get\_dummies(X)

```

```python

diab\_LSR = pd.concat((X,y), axis=1)

```

```python

diab\_LSR = pd.DataFrame(diab\_LSR)

```

```python

diab\_LSR.info(1)

```

```python

X= X.values

y= y.values

```

```python

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression(max\_iter=10000)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.35, shuffle=True, stratify=y,random\_state=123)

```

```python

from sklearn.model\_selection import cross\_val\_score

for i in np.logspace(-10,9,19,10):

lr.C = i

out = cross\_val\_score(lr,X\_train,y\_train,scoring='accuracy', cv=5)

print(out.mean(), out.std(), i)

```

```python

lr = LogisticRegression(max\_iter=10000,C = 0.0278)

Readmiss\_pred = lr.fit(X\_train,y\_train)

Readmiss\_pred.score(X\_test,y\_test)

```

```python

from sklearn.model\_selection import cross\_val\_predict

pred = cross\_val\_predict(lr,X\_test,y\_test, cv=5)

```

```python

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,pred,

target\_names=['No\_readmission', 'readmission <30days', 'readmission >30days']))

```

```python

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

cm0 = confusion\_matrix(y\_test,pred)

x\_axis\_labels = ['NO','<30', '>30']

y\_axis\_labels = ['NO','<30', '>30']

ax = plt.axes()

sns.heatmap(cm0, cmap='Blues', annot=True, fmt='d',xticklabels=x\_axis\_labels,

yticklabels=y\_axis\_labels, ax=ax, cbar\_kws={'label': 'Readmission Count',})

ax.set\_title('Hospital Readmission Confusion Matrix Heatmap')

plt.xlabel("Predicted readmission")

plt.ylabel("True readmission")

plt.show

```

```python

cm0

```

```python

#Feature Importance

feature\_names = diab\_LSR.drop('readmitted', axis=1).copy().columns.values

feature\_importances = pd.DataFrame(Readmiss\_pred.coef\_[0],

index = feature\_names,

columns=['importance']).sort\_values('importance',

ascending=False)

```

```python

feature\_importances

```

```python

#Feature Importance plot

num = 50

ylocs = np.arange(num)

# get the feature importance for top num and sort in reverse order

values\_to\_plot = feature\_importances.iloc[:num].values.ravel()[::-1]

feature\_labels = list(feature\_importances.iloc[:num].index)[::-1]

plt.figure(num=None, figsize=(8, 15), dpi=80, facecolor='w', edgecolor='k');

plt.barh(ylocs, values\_to\_plot, align = 'center')

plt.ylabel('Features', fontsize=20)

plt.xlabel('Importance Score', fontsize=20)

plt.title('Feature Importance Score - Readmission Classification, Logistic Regression', fontsize=20)

plt.yticks(ylocs, feature\_labels, fontsize=11)

plt.show()

```