Case Study 2: Medical Data

Tim Cabaza

May 31st, 2024

**Abstract**

The case study is an exercise in building a classifier using Logistic Regression to predict a multi-class target variable (hospital readmittance). The following is a list of steps that are covered in the case study: Exploratory Data Analysis, Data Cleansing, Imputation, Label Encoding, Normalizing, Logisitic Regression, and KFold Cross Validation.

**1 Introduction**

The data was collected over a 10 year period with over 130 hospitals; given the different entry methodologies by each hospital there are holes in the data that will be addressed using the most reasonable imputation methods. The goal of this exercise is to create the most accurate classification model possible, identifying the best regularization parameter (C), and the top five most important features.

**2 Methods**

**Exploratory Data Analysis:** The data has 50 columns of data with 101,766 values. The target variable for this case study will be the column readmitted.

Before data cleansing, the data was inspected feature by feature to understand the missing values, dtypes, and general structure of the data.

**Table 1: Original Data**

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Column Name | dtype | Comments on data |
| 0. | encounter\_id | int64 | * discrete variable * no missing values |
|  | patient\_nbr | int64 | * discrete variable * no missing values |
|  | race | object | * change dtype to categorical * label encode 5 classes * 2270 missing values at random * impute or drop column * ethical questions regarding the use of race * impute missing data with mode |
|  | gender | object | * multi-class categorical * label encode 3 classes * no missing values |
|  | age | object | * multi-class categorical * label encode 10 classes * no missing values |
|  | weight | object | * multi-class categorical variable * label encode 9 classes * missing values 97% of its value |
|  | admission\_type\_id | int64 | * discrete * label encode 8 classes   somewhat correlated with num\_medications |
|  | discharge\_disposition\_id | int64 | * discrete * label encode 26 classes |
|  | admission\_source\_id | int64 | * discrete * label encode 17 classes |
|  | time\_in\_hospital | int64 | * discrete * label encode 14 classes |
|  | payer\_code | object | * Change dtype to categorical * label encode 17 classes * missing values |
|  | medical\_specialty | object | * Change dtype to categorical * label encode 72 classes * missing values |
|  | num\_lab\_procedures | int64 | * discrete * label encode 118 classes |
|  | num\_procedures | int64 | * discrete * label encode 7 classes |
|  | num\_medications | int64 | * discrete * label encode 75 classes |
|  | number\_outpatient | int64 | * discrete * label encode 39 classes |
|  | number\_emergency | int64 | * discrete * label encode 33 classes |
|  | number\_inpatient | int64 | * discrete * label encode 21 classes |
|  | diag\_1 | int64 | * Change dtype to categorical * label encode 716 classes |
|  | diag\_2 | int64 | * Change dtype to categorical * label encode 748 classes |
|  | diag\_3 | int64 | * Change dtype to categorical * label encode 789 classes |
|  | number\_diagnoses | int64 | * discrete |
|  | max\_glu\_serum | object | * Change dtype to categorical * label encode 3 classes |
|  | A1Cresult | object | * Change dtype to categorical * label encode 4 classes * missing values aka None |
|  | metformin | object | * Change dtype to categorical * label encode 4 classes |
|  | repaglinide | object | * Change dtype to categorical * label encode 4 classes |
|  | nateglinide | object | * Change dtype to categorical * label encode 4 classes |
|  | chlorpropamide | object | * Change dtype to categorical * label encode 4 classes |
|  | glimepiride | object | * Change dtype to categorical * label encode 4 classes |
|  | acetohexamide | object | * Change dtype to categorical * label encode 4 classes |
|  | glipizide | object | * Change dtype to categorical * label encode 4 classes |
|  | glyburide | object | * Change dtype to categorical * label encode 4 classes |
|  | tolbutamide | object | * Change dtype to categorical * label encode 2 classes |
|  | pioglitazone | object | * Change dtype to categorical * label encode 4 classes |
|  | rosiglitazone | object | * Change dtype to categorical * label encode 4 classes |
|  | acarbose | object | * Change dtype to categorical * label encode 4 classes |
|  | miglitol | object | * Change dtype to categorical * label encode 4 classes |
|  | troglitazone | object | * Change dtype to categorical * label encode 2 classes |
|  | tolazamide | object | * Change dtype to categorical * label encode 3 classes |
|  | examide | object | * Change dtype to categorical * label encode 1 classes |
|  | citoglipton | object | * Change dtype to categorical * label encode 1 classes |
|  | insulin | object | * Change dtype to categorical * label encode 4 classes |
|  | glyburide-metformin | object | * Change dtype to categorical * label encode 2 classes |
|  | glipizide-metformin | object | * Change dtype to categorical * label encode 2 classes |
|  | glimepiride-pioglitazone | object | * Change dtype to categorical * label encode 2 classes |
|  | metformin -rosiglitazone | object | * Change dtype to categorical * label encode 2 classes |
|  | metformin-pioglitazone | object | * Change dtype to categorical * label encode 2 classes |
|  | change | object | * Change dtype to categorical * label encode 2 classes |
|  | diabetesMed | object | * Change dtype to categorical * label encode 2 classes |
|  | readmitted \*target\* | object | * Change to categorical variable * label encode 3 classes |

The target variable, readmitted has three distinct classes and was label encoded to create the model

**A blue rectangular object with numbers

Description automatically generated**

**Figure (1):** target variable – readmitted

A multi-class variable with **54%** *NO*, **35%** *>30*, and **11%** *<30*

The data was analyzed for missing data. There were a number of missing values denoted with question marks, the question marks were converted NAN so that Python could readily see the missing values.

Weight has nearly 97% missing data and is dropped from the dataset. Generally its best practice to avoid dropping columns but given the vast quantity of missing data, there is little info to properly impute.

A screenshot of a number

Description automatically generated

**Figure (2):** weight variable

Dropped from dataset, too many missing values to impute.

Race was the first column with 2270 columns missing. The data was imputed using the mode (the most common) category. By using the most common value (mode) we avoid putting any undue influence on the data, as no discernable pattern was found. Regarding the use of such variable in the model is dependent on the use objective of the customer. In many cases race as a category can open questions into bias and racism in a study, however given that this is a health-related case study we will proceed with caution and use the variable in our model.

A screenshot of a cell phone

Description automatically generated

**Figure (3):** race variable

Missing at random, imputed the most common value (mode).

A blue and white graph

Description automatically generated

**Figure (4):** race variable

Ethically this is a questionable variable to include, however given the nature of study being medically related we will impute the mode in this case ‘Caucasian’ for missing values.

The payer\_code another variable with 40% missing values. While the feature was somewhat correlated with encounter\_id with a 0.56 correlation ratio coefficient, the lack of discernable pattern meant that the data was most likely missing at random. 60% of the data is okay to use, so given that it would be best not to drop the variable, instead another class is added ‘unknown’ as this was already a 17 class variable before (now 18).

A screenshot of a computer code

Description automatically generated

**Figure (5):** payer\_code variable

Missing at random, imputed a new label ‘unknown’ to preserve the 60% of data that was present.

The medical\_specialty variable has 49% missing values with 72 classes. No discernable pattern when comparing with other features could be found and therefore the imputation strategy was to add a new class ‘unknown’ to the multi-class categorical feature.

A screenshot of a medical specialty

Description automatically generated

**Figure (5):** payer\_code variable

Missing at random, imputed a new label ‘unknown’ to preserve the 60% of data that was present.

A1Cresult had a class with None values, but given None is a special constant in Python, the values were replaced with N.

A screenshot of a computer

Description automatically generated

**Figure (6):** A1Cresult had

Missing at random, imputed a new label ‘unknown’ to preserve the 60% of data that was present.

Max\_glu\_serum had a class of None values, but given that None is a special constant in Python, the values were replaced with N.

A screenshot of a computer

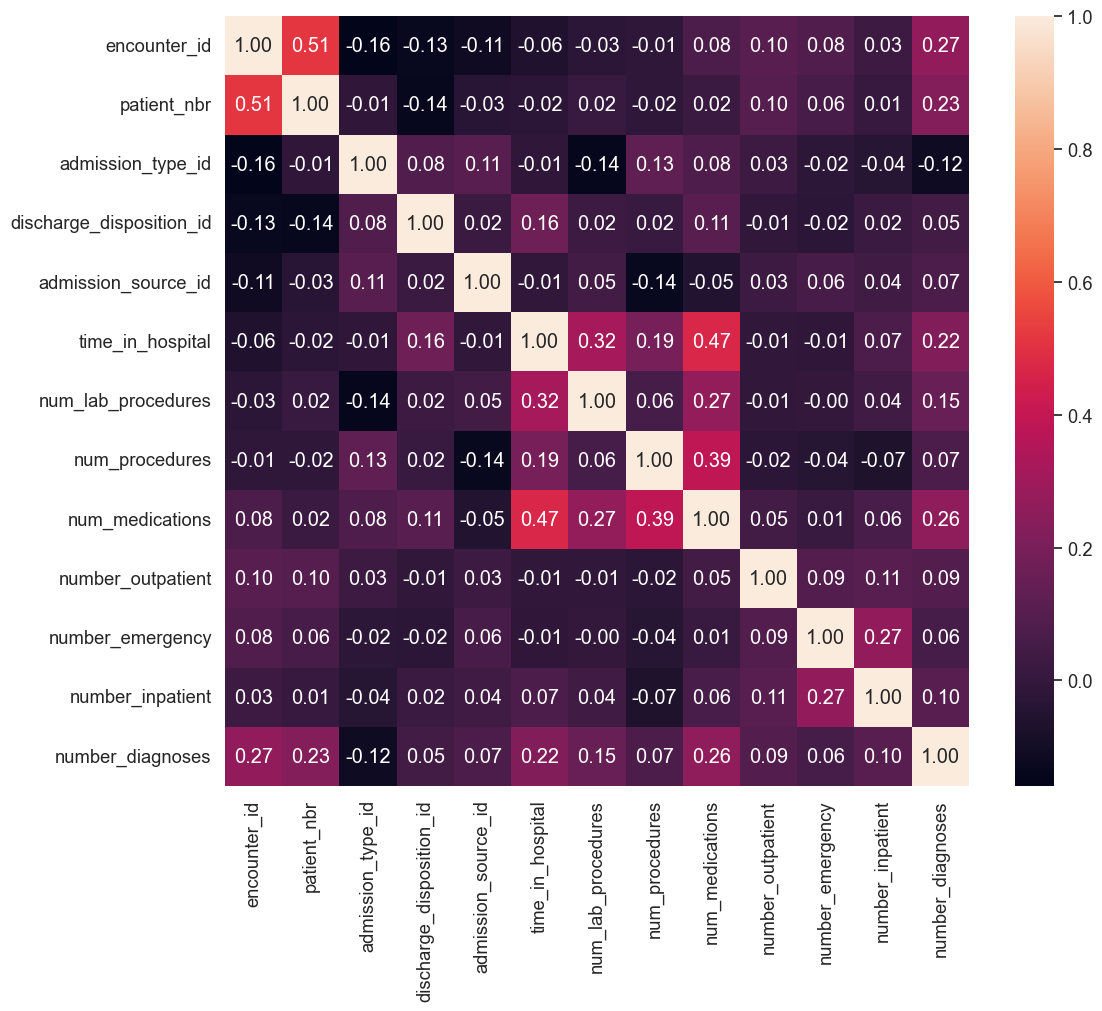
Description automatically generated

**Figure (7):** max\_glu\_serum variable

Imputed a new label ‘unknown’ to preserve the 60% of data that was present.

After the imputation was complete, the datatypes were addressed. Features diag\_1, diag\_2, diag\_3 had to be converted to a multi-class categorical variable as it is registering as a int64 object and this is not reflective of the data as it is not discrete. All other datatypes seemed appropriate. The rest of the object datatype features were converted to a categorical and label encoded using sklearn’s LabelEncoder including the target variable which consists of 3 classes. The numerical values were left as is (int64).

After imputation and dropping of the column weight, the dataset had a shape of (101766, 49) including our target variable ‘readmitted.



**Figure (8):** Pairwise correlation matrix between numeric data.

Patient\_nbr and encounter\_id had the highest pairwise correlation with any variable at 0.512028. This is not considered highly correlated but somewhat correlated.

**Table 2**

**Final Dataset for Modeling (Cleansed, Imputed, Normalized)**

**Note: abbreviated to save space, refer to code for complete charts for all features.**

|  |  |  |
| --- | --- | --- |
| Index | Column Name | Plot |
| 0 | encounter\_id |  |
| 2 | patient\_nbr |  |
| 3 | race |  |
| 4 | gender |  |
| 5 | age |  |
| 10 | payer\_code |  |
| 11 | medical\_specialty |  |
| 18 | diag\_1 |  |
| 19 | diag\_2 |  |
| 20 | diag\_3 |  |
| 22 | max\_glum\_serum |  |
| 23 | A1Cresult: |  |

**3 Results**

The dataset once readied to be modeled, sklearns Logistic RegressionCV with OVR (one-versus-rest) was done over a logspace (adjusted and tested had to use smaller values) to find the optimal regularization value (c). The best score (f1\_macro) was collected as well as the following Classification Report and Confusion Matrix Report were generated.

**Table 3**

**The Top 5 Features**

|  |  |
| --- | --- |
| Best Regurlarization Parameter : | 0.8858667904100823 |
| Top 5 Features: | * 'number\_emergency' * 'number\_diagnoses' * 'patient\_nbr' * 'encounter\_id', * 'number\_inpatient' |
| Best cross-validation F1 macro score: | 0.486207403452041 |
| Classification Report: |  |
| Confusion Matrix Report |  |

**4 Conclusions**

The case study assessed a medical dataset. There were a number of missing values and None type values that had to be either, dropped or imputed on a case by case basis. Once the data was cleansed, data types set, label-encoded, and normalized the Logisitc Regression CV from Sklearn was utilized. All data other than weight, was kept despite possible ethical questions regarding race, age, and gender. In this case, the data is used for medical reasons and to gain insights and the customer has expressed that they believe the data is important enough due to possible links, therefore it was cautiously included. However, outside the scope of the medical field given the sensitivity of the data, it would be good practice to reassess the use of these particular columns of data.

The model was then tuned for the best regularization parameter 0.8858667904100823returned an F1 score of 0.486207403452041. The top 5 most important features were: ‘number\_emergency’, ‘number\_diagnoses’, ‘patient\_nbr’, ‘ecnounter\_id’, ‘number\_inpatient’.

**Appendix: Code**

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import Ridge

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

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import GridSearchCV

# from tabulate import tabulate

import pandas as pd

import numpy as np

# !pip install sweetviz

import sweetviz as sv

# The data was downloaded from the SMU \_\_\_ website and then the file paths for both files are a assigned a variable (filepath and filepath2).

filepath = "/Users/tmc/Desktop/MS\_SMU\_Admin/05\_2024Summer/QTW/Data\_Science/Case\_Study2/dataset\_diabetes/diabetic\_data.csv" # one dot current directory, two dots means the parent directory one level up from the current directory

filepath2 = "/Users/tmc/Desktop/MS\_SMU\_Admin/05\_2024Summer/QTW/Data\_Science/Case\_Study2/dataset\_diabetes/IDs\_mapping.csv"

df\_ref = pd.DataFrame(pd.read\_csv(filepath2))

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.max\_rows', None)

df\_ref.head(20) # now all rows

df\_ref.head(len(df\_ref)) # all rows

df\_ref.fillna('---', inplace=True)

df\_ref.head(len(df\_ref))

df\_original = pd.DataFrame(pd.read\_csv(filepath))

df\_original.head()

# dataframe with df columns

print(f'df\_original shape:', df\_original.shape)

pd.DataFrame(df\_original.columns, columns=['columns'])

df\_original.dtypes

print(df\_original.info())

# Create another dataframe to work with

df = df\_original.copy()

df.head()

import pandas as pd

unique\_values = pd.DataFrame(columns=['Column', 'Unique Values'])

for col in df.columns:

nunique = df[col].nunique()

value\_counts = df[col].value\_counts().to\_string()

new\_df = pd.DataFrame({'Column': [col], 'Unique Values': [nunique]}) #

unique\_values = pd.concat([unique\_values, new\_df], ignore\_index=True)

# check cardinality of the first 25

unique\_values.sort\_values(by='Unique Values', ascending=False)

# unique\_values.sort\_values(by='Unique Values', ascending=False).head(25)

# unique\_values.sort\_values(by='Unique Values', ascending=False).tail(25)

print(df.columns)

# Replace the '?' with NaN

cols = ['race', 'weight', 'discharge\_disposition\_id', 'payer\_code', 'medical\_specialty', 'A1Cresult', 'max\_glu\_serum']

df[cols] = df[cols].replace("?", np.nan)

df[cols] = df[cols].replace("None", np.nan)

df.head()

# Generate the report using SweetViz package, it takes a few minutes to run but provides a comprehensive report on the data

import sweetviz as sv

report = sv.analyze(df\_original)

report.show\_notebook()

# Check for missing values by column

missing\_values = pd.DataFrame(df.isnull().sum(), columns=['Missing Values'])

missing\_values

df['payer\_code'].sort\_values().unique()

# make a pd.dataframe wiht the array

pd.DataFrame(df['payer\_code'].sort\_values().unique())

# fix dtype on diag\_1, diag\_2, diag\_3

df['diag\_1'] = df['diag\_1'].astype(object)

df['diag\_2'] = df['diag\_2'].astype(object)

df['diag\_3'] = df['diag\_3'].astype(object)

df.describe()

# column race has 2% missing values, replace with mode

# Get the mode (most frequent value)

mode = df['race'].mode().iloc[0]

print(mode)

df['race'] = df['race'].fillna(mode)

# Print the info for the 'race' column

print(df[['race']].info())

df['race'].isnull().sum()

# Nearly 40% of payer code is missing. Replace with 'unknown'

df['payer\_code'] = df['payer\_code'].fillna('unknown')

print(df['payer\_code'].unique())

print(df[['payer\_code']].info())

df['payer\_code'].isnull().sum()

# Nearly 47% of medical specialty is missing. Replace with 'unknown'

df['medical\_specialty'] = df['medical\_specialty'].fillna('unknown')

print(df['medical\_specialty'].unique())

print(df[['medical\_specialty']].info())

# check for missing values on medical\_specialty

df['medical\_specialty'].isnull().sum()

# None is a special constant in python and is not seen as a class in max\_glu\_serum, create a class with 'unknown'

df['max\_glu\_serum'] = df['max\_glu\_serum'].fillna('N')

print(df['max\_glu\_serum'].unique())

print(df[['max\_glu\_serum']].info())

# check for missing values on medical\_specialty

df['max\_glu\_serum'].isnull().sum()

# None is a special constant in python and is not seen as a class in A1Cresult, create a class with 'unknown'

df['A1Cresult'] = df['A1Cresult'].fillna('N')

print(df['A1Cresult'].unique())

print(df[['A1Cresult']].info())

# check for missing values on medical\_specialty

df['A1Cresult'].isnull().sum()

df.isnull().sum()

# now lets drop the column weight and rename the dataset

df\_new = df.drop(columns=['weight'])

df\_new.head()

print(df\_new.columns)

print(df\_new.shape)

import sweetviz as sv

from IPython.display import IFrame

report = sv.analyze(df\_new)

# Run report again to see the missing values

report.show\_notebook()

from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt

# Select object columns

object\_columns = df\_new.select\_dtypes(include='object').columns

# Create DataFrame of object columns

df\_object\_columns = pd.DataFrame(object\_columns, columns=['object columns'])

# Convert all to categorical and print unique values, value counts, and bar chart

for col in object\_columns:

df\_new[col] = df\_new[col].astype('category')

print(df\_new[col].unique())

print(df\_new[col].value\_counts())

df\_new[col].value\_counts().plot(kind='bar')

plt.show()

# Initialize LabelEncoder

labelencoder = LabelEncoder()

df\_new[col] = labelencoder.fit\_transform(df\_new[col])

import seaborn

import matplotlib.pyplot as plt

# get the correlation of the columns

sns.set(font\_scale=1.2)

plt.figure(figsize=(12, 10))

sns.heatmap(df\_new.select\_dtypes(include='float64').corr(), annot=True, fmt=".2f")

# pateint\_nbr 0.51 correlation with encounter\_id, is the highest correlation value

# get the highest correlation value

print(df\_new.select\_dtypes(include='float64').corr().unstack().sort\_values(ascending=False).drop\_duplicates())

analzye = sv.analyze(df\_new)

analzye.show\_notebook()

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegressionCV

from sklearn.model\_selection import KFold

import numpy as np

from sklearn.metrics import classification\_report

from sklearn.metrics import ConfusionMatrixDisplay

# Assume df\_new is your DataFrame and 'readmitted' is the target column

# Ensure 'readmitted' is categorical

df\_new['readmitted'] = df\_new['readmitted'].astype('category')

# Encode 'readmitted' as integers

le = LabelEncoder()

df\_new['readmitted'] = le.fit\_transform(df\_new['readmitted'])

# Loop over all categorical columns and encode them

for col in df\_new.select\_dtypes(include=['category']).columns:

if col != 'readmitted': # Skip 'readmitted' as it's already encoded

df\_new[col] = le.fit\_transform(df\_new[col])

# Separate features and target

X = df\_new.drop(['readmitted'], axis=1)

y = df\_new['readmitted']

# Scale features

scaler = StandardScaler()

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

# Define the parameter space for C

C = np.logspace(-3, 1, 20)

splits = KFold(n\_splits=5, shuffle=True, random\_state=42)

# Create the LogisticRegressionCV model for One-vs-Rest (OVR) classification

model = LogisticRegressionCV(Cs=C, cv=splits, max\_iter=1000, scoring='f1\_macro', multi\_class='ovr')

model.fit(X\_scaled, y)

# Retrieve the best regularization parameter

best\_C = model.C\_[0]

print("Best C:", best\_C)

importance = model.coef\_

# Sum the absolute values of the coefficients across all classes for feature importance

abs\_importance = np.sum(np.abs(importance), axis=0)

# Get the indices of the top 5 features

top\_5\_idx = np.argsort(abs\_importance)[-5:]

# Get the names of the top 5 features

top\_5\_features = X.columns[top\_5\_idx]

print("Top 5 features:", top\_5\_features)

# Display the shapes for verification

print("Data shape:", df\_new.shape)

print("Coefficients shape:", model.coef\_.shape)

print("Intercept:", model.intercept\_)

best\_score = model.scores\_[1].max()

print("Best cross-validation F1 macro score:", best\_score)

predictions = model.predict(X\_scaled)

print(classification\_report(y, predictions))

# Plot the confusion matrix

ConfusionMatrixDisplay.from\_predictions(y, predictions, cmap='Greens')

plt.show()