Quantifying the World

Case Study 2

Presented by:

Alexander Sepenu

Nnenna Okpara

Taifur Chowdhury

Juan Edgar Nunez - Gonzalez

Business Understanding

Quantifying the World

Objective: The objective of this case study is to build a classification model using logistic regression which predicts hospital readmittance.

In [1]:
```
import os
os.chdir(r'/Users/juannunez/Documents/SMU/Quantifying The World/Case study 2'
```

In [2]:
```
import pandas as pd
import numpy as np
import pandas_profiling
from pandas_profiling import ProfileReport
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [3]:
```
# reading in the data
dai_df = pd.read_csv('diabetic_data.csv')
```

In [4]:

```python
# visualizing the shape of the data for rows and column
print("Diabetes Data Shape is: {}".format(dai_df.shape))
```

```
Diabetes Data Shape is: (101766, 50)
```

Data Evaluation / Engineering

Exploratory Data Analysis

Summary

The data contains missing values which are represented with a question mark. Hence, we looked for where and how many values are missing in the dataset. To facilitate the analysis of this data we will replace the "?" with a NaN.

Next we identified columns with diagnostic codes (i.e. "diag_1", diag_2", "diag_3") that are mostly numeric, with some exceptions where the code is alphanumeric. Since, we don't have a way to identify the meaning of the diagnostic codes, we elected to remove the alphanumeric anomalies (i.e., “V10”), etc.; Because these values can affect the entire Logistic Regression, if not corrected.

In the third step, we computed the percentages of missing values in every column. Our team considered to drop the columns where more than 20% of the values were missing, and impute the missing values in columns where less than 10% of the values were missing. However, with since there is no absolute rule of thumb, we settled on a more conservative approach based on James Ledoux https://jamesrledoux.com/code/imputation, experience and dropped any column with more than 5% of values missing and imputed columns with less than 5% of values missing.

We use the Mode to impute the missing values because it is the value less likely to skew the data.

In [5]:

```python
# Looking at the first column of data to see the distribution of data across
dai_df.head()
```

Out[5]:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discha
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	
1	149190	55629189	Caucasian	Female	[10-20)	?	1	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	
3	500364	82442376	Caucasian	Male	[30-40)	?	1	
4	16680	42519267	Caucasian	Male	[40-50)	?	1	

5 rows × 50 columns

In [6]:
```
# changing "?" character to NAN to have an over view of missing values within

dai_df = dai_df.replace('?',np.nan)
```

In [7]:
```
# looking at the dataset data type

dai_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 50 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   encounter_id              101766 non-null  int64 
 1   patient_nbr               101766 non-null  int64 
 2   race                      99493 non-null   object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    3197 non-null    object
 6   admission_type_id         101766 non-null  int64 
 7   discharge_disposition_id  101766 non-null  int64 
 8   admission_source_id       101766 non-null  int64 
 9   time_in_hospital          101766 non-null  int64 
 10  payer_code                61510 non-null   object
 11  medical_specialty         51817 non-null   object
 12  num_lab_procedures        101766 non-null  int64 
 13  num_procedures            101766 non-null  int64 
 14  num_medications           101766 non-null  int64 
 15  number_outpatient         101766 non-null  int64 
 16  number_emergency          101766 non-null  int64 
 17  number_inpatient          101766 non-null  int64 
 18  diag_1                    101745 non-null  object
 19  diag_2                    101408 non-null  object
 20  diag_3                    100343 non-null  object
 21  number_diagnoses          101766 non-null  int64
 22  max_glu_serum             101766 non-null  object
 23  A1Cresult                 101766 non-null  object
 24  metformin                 101766 non-null  object
 25  repaglinide               101766 non-null  object
 26  nateglinide               101766 non-null  object
 27  chlorpropamide            101766 non-null  object
 28  glimepiride               101766 non-null  object
 29  acetohexamide             101766 non-null  object
 30  glipizide                 101766 non-null  object
 31  glyburide                 101766 non-null  object
 32  tolbutamide               101766 non-null  object
 33  pioglitazone              101766 non-null  object
 34  rosiglitazone             101766 non-null  object
 35  acarbose                  101766 non-null  object
 36  miglitol                  101766 non-null  object
 37  troglitazone              101766 non-null  object
 38  tolazamide                101766 non-null  object
 39  examide                   101766 non-null  object
 40  citoglipton               101766 non-null  object
 41  insulin                   101766 non-null  object
 42  glyburide-metformin       101766 non-null  object
 43  glipizide-metformin       101766 non-null  object
 44  glimepiride-pioglitazone  101766 non-null  object
 45  metformin-rosiglitazone   101766 non-null  object
 46  metformin-pioglitazone    101766 non-null  object
 47  change                    101766 non-null  object
 48  diabetesMed               101766 non-null  object
 49  readmitted                101766 non-null  object
dtypes: int64(13), object(37)
memory usage: 38.8+ MB
```

In [8]:
```
# Data set head with NaN to see the distribution of missing values in data
dai_df.head()
```

Out[8]:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discha
0	2278392	8222157	Caucasian	Female	[0-10)	NaN	6	
1	149190	55629189	Caucasian	Female	[10-20)	NaN	1	
2	64410	86047875	AfricanAmerican	Female	[20-30)	NaN	1	
3	500364	82442376	Caucasian	Male	[30-40)	NaN	1	
4	16680	42519267	Caucasian	Male	[40-50)	NaN	1	

5 rows × 50 columns

In [9]:

```
# Fishing out the number of missing values per column as part of the EDA

dai_df.isnull().sum()
```

Out[9]:
```
encounter_id                    0
patient_nbr                     0
race                         2273
gender                          0
age                             0
weight                      98569
admission_type_id               0
discharge_disposition_id        0
admission_source_id             0
time_in_hospital                0
payer_code                  40256
medical_specialty           49949
num_lab_procedures              0
num_procedures                  0
num_medications                 0
number_outpatient               0
number_emergency                0
number_inpatient                0
diag_1                         21
diag_2                        358
diag_3                       1423
number_diagnoses                0
max_glu_serum                   0
A1Cresult                       0
metformin                       0
repaglinide                     0
nateglinide                     0
chlorpropamide                  0
glimepiride                     0
acetohexamide                   0
glipizide                       0
glyburide                       0
tolbutamide                     0
pioglitazone                    0
rosiglitazone                   0
acarbose                        0
miglitol                        0
troglitazone                    0
tolazamide                      0
examide                         0
citoglipton                     0
insulin                         0
glyburide-metformin             0
glipizide-metformin             0
glimepiride-pioglitazone        0
metformin-rosiglitazone         0
metformin-pioglitazone          0
change                          0
diabetesMed                     0
readmitted                      0
dtype: int64
```

In [10]:

```
# Using regex function to correct wrongly entered values for diag_1, diag_2, 

dai_df["diag_1"] =dai_df["diag_1"].replace(to_replace ='[V]', value = '', reg
dai_df["diag_1"] = dai_df["diag_1"].replace(to_replace ='[E]', value = '', re

dai_df["diag_2"] =dai_df["diag_2"].replace(to_replace ='[V]', value = '', reg
dai_df["diag_2"] = dai_df["diag_2"].replace(to_replace ='[E]', value = '', re

dai_df["diag_3"] =dai_df["diag_3"].replace(to_replace ='[V]', value = '', reg
dai_df["diag_3"] = dai_df["diag_3"].replace(to_replace ='[E]', value = '', re
```

In [11]:

```
# Calculating the percentages of missing values per column to direct our impu
((dai_df.isnull() | dai_df.isna()).sum() * 100 / dai_df.index.size).round(2)
```

Out[11]:
```
encounter_id                 0.00
patient_nbr                  0.00
race                         2.23
gender                       0.00
age                          0.00
weight                      96.86
admission_type_id            0.00
discharge_disposition_id     0.00
admission_source_id          0.00
time_in_hospital             0.00
payer_code                  39.56
medical_specialty           49.08
num_lab_procedures           0.00
num_procedures               0.00
num_medications              0.00
number_outpatient            0.00
number_emergency             0.00
number_inpatient             0.00
diag_1                       0.02
diag_2                       0.35
diag_3                       1.40
number_diagnoses             0.00
max_glu_serum                0.00
A1Cresult                    0.00
metformin                    0.00
repaglinide                  0.00
nateglinide                  0.00
chlorpropamide               0.00
glimepiride                  0.00
acetohexamide                0.00
glipizide                    0.00
glyburide                    0.00
tolbutamide                  0.00
pioglitazone                 0.00
rosiglitazone                0.00
acarbose                     0.00
miglitol                     0.00
troglitazone                 0.00
tolazamide                   0.00
examide                      0.00
citoglipton                  0.00
insulin                      0.00
glyburide-metformin          0.00
glipizide-metformin          0.00
glimepiride-pioglitazone     0.00
metformin-rosiglitazone      0.00
metformin-pioglitazone       0.00
change                       0.00
diabetesMed                  0.00
readmitted                   0.00
dtype: float64
```

In [12]:
```
#dropping unwanted columns with large missing values
new_diab = dai_df.drop(['weight','payer_code','medical_specialty' ], axis = 1
```

In [13]:
```
# imputing nan vallues
new_diab = new_diab.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

In [14]:
```
#Confirming that there are no missing values
((new_diab.isnull() | new_diab.isna()).sum() * 100 / new_diab.index.size).rou
```

Out[14]:
```
encounter_id                0.0
patient_nbr                 0.0
race                        0.0
gender                      0.0
age                         0.0
admission_type_id           0.0
discharge_disposition_id    0.0
admission_source_id         0.0
time_in_hospital            0.0
num_lab_procedures          0.0
num_procedures              0.0
num_medications             0.0
number_outpatient           0.0
number_emergency            0.0
number_inpatient            0.0
diag_1                      0.0
diag_2                      0.0
diag_3                      0.0
number_diagnoses            0.0
max_glu_serum               0.0
A1Cresult                   0.0
metformin                   0.0
repaglinide                 0.0
nateglinide                 0.0
chlorpropamide              0.0
glimepiride                 0.0
acetohexamide               0.0
glipizide                   0.0
glyburide                   0.0
tolbutamide                 0.0
pioglitazone                0.0
rosiglitazone               0.0
acarbose                    0.0
miglitol                    0.0
troglitazone                0.0
tolazamide                  0.0
examide                     0.0
citoglipton                 0.0
insulin                     0.0
glyburide-metformin         0.0
glipizide-metformin         0.0
glimepiride-pioglitazone    0.0
metformin-rosiglitazone     0.0
metformin-pioglitazone      0.0
change                      0.0
diabetesMed                 0.0
readmitted                  0.0
dtype: float64
```

In [15]:
```
# Visualizing basic statitics for the data
new_diab.describe()
```

Out[15]:

	encounter_id	patient_nbr	admission_type_id	discharge_disposition_id	admission_s
count	1.017660e+05	1.017660e+05	101766.000000	101766.000000	101766
mean	1.652016e+08	5.433040e+07	2.024006	3.715642	5
std	1.026403e+08	3.869636e+07	1.445403	5.280166	4
min	1.252200e+04	1.350000e+02	1.000000	1.000000	1
25%	8.496119e+07	2.341322e+07	1.000000	1.000000	1
50%	1.523890e+08	4.550514e+07	1.000000	1.000000	7
75%	2.302709e+08	8.754595e+07	3.000000	4.000000	7
max	4.438672e+08	1.895026e+08	8.000000	28.000000	25

Detail EDA with vizualization

We are recoding all categorical variables into numeric to do a thorough EDA and feature analysis.

In [16]:
```
# convert categorical to integers
cleanup_nums = {"race": {"AfricanAmerican": 0, "Asian": 1, "Caucasian": 2, "H
                "gender": {"Female": 0, "Male": 1, "Unknown/Invalid": 2},
                "age": {"[0-10)":0, "[10-20)":1, "[20-30)":2, "[30-40)":3, "[
                "max_glu_serum": {">200": 0, ">300": 1, ">300": 2, "None": 3,
                "A1Cresult": {">7": 0, ">8": 1, "None": 2, "Norm": 4},
                "metformin": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "repaglinide": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "nateglinide": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "chlorpropamide": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "glimepiride": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "acetohexamide": {"No": 0, "Steady": 1},
                "glipizide": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "glyburide": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "tolbutamide": {"No": 0, "Steady": 1},
                "pioglitazone": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "rosiglitazone": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "acarbose": {"Down":0, "No":1, "Steady":2, "Up":3},
                "miglitol": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "troglitazone": {"No": 0, "Steady": 1},
                "tolazamide": {"No": 0, "Steady": 1, "Up": 2},
                "examide":{"No": 0},
                "citoglipton": {"Down":0, "No":1, "Steady":2, "Up":3},
                "insulin": {"Down": 0, "No": 1, "Steady": 2, "Up": 3},
                "glyburide-metformin": {"Down":0, "No":1, "Steady":2, "Up":3}
                "glipizide-metformin": {"No": 0, "Steady": 1},
                "glimepiride-pioglitazone": {"Down":0, "No":1, "Steady":2, "U
                "metformin-rosiglitazone": {"No": 0, "Steady": 1},
                "metformin-pioglitazone": {"No": 0, "Steady": 1},
                "change": {"No":0, "Ch":1},
                "diabetesMed":{"No": 0, "Yes": 1}}
```

In [17]:
```
#Integrating all recorded numeric features into dataframes
new_diab = new_diab.replace(cleanup_nums)
```

In [18]:
```
#Changing float data type into intergers
new_diab['diag_1'] =new_diab['diag_1'].astype(float).astype(int)
new_diab['diag_2'] = new_diab['diag_2'].astype(float).astype(int)
new_diab['diag_3'] = new_diab['diag_3'].astype(float).astype(int)
new_diab['gender'] = new_diab['gender'].astype(int)
```

In [19]:
```
target_labels = new_diab['readmitted'].unique().tolist()
```

In [20]:
```
target_mod = {
        'NO':0,
        '>30':0,
        '<30':1
}
```

In [21]:
```
new_diab['readmitted_binary'] = new_diab['readmitted'].map(target_mod)
new_diab['readmitted_binary'].unique()
```

Out[21]:
```
array([0, 1])
```

In [22]:
```
#dropping unwanted columns
new_diab = new_diab.drop(['readmitted'], axis = 1)
new_diab = new_diab.drop(['examide'], axis = 1)
new_diab = new_diab.drop(['citoglipton'], axis = 1)
```

In [24]:
```
#Dropping target feature for feature analysis
new_diab_df = new_diab.drop(['readmitted_binary'], axis = 1)
```

In [25]:
```
new_diab_df.dtypes
```

Out[25]:

```
encounter_id                int64
patient_nbr                 int64
race                        int64
gender                      int64
age                         int64
admission_type_id           int64
discharge_disposition_id    int64
admission_source_id         int64
time_in_hospital            int64
num_lab_procedures          int64
num_procedures              int64
num_medications             int64
number_outpatient           int64
number_emergency            int64
number_inpatient            int64
diag_1                      int64
diag_2                      int64
diag_3                      int64
number_diagnoses            int64
max_glu_serum               int64
A1Cresult                   int64
metformin                   int64
repaglinide                 int64
nateglinide                 int64
chlorpropamide              int64
glimepiride                 int64
acetohexamide               int64
glipizide                   int64
glyburide                   int64
tolbutamide                 int64
pioglitazone                int64
rosiglitazone               int64
acarbose                    int64
miglitol                    int64
troglitazone                int64
tolazamide                  int64
insulin                     int64
glyburide-metformin         int64
glipizide-metformin         int64
glimepiride-pioglitazone    int64
metformin-rosiglitazone     int64
metformin-pioglitazone      int64
change                      int64
diabetesMed                 int64
dtype: object
```

In [ ]:

```
#(new_diab_df.profile_report()).to_file('Diabetes.html')
```

In [27]:

```
#diab_profile = ProfileReport(new_diab, title="Profiling Report", explorative
```

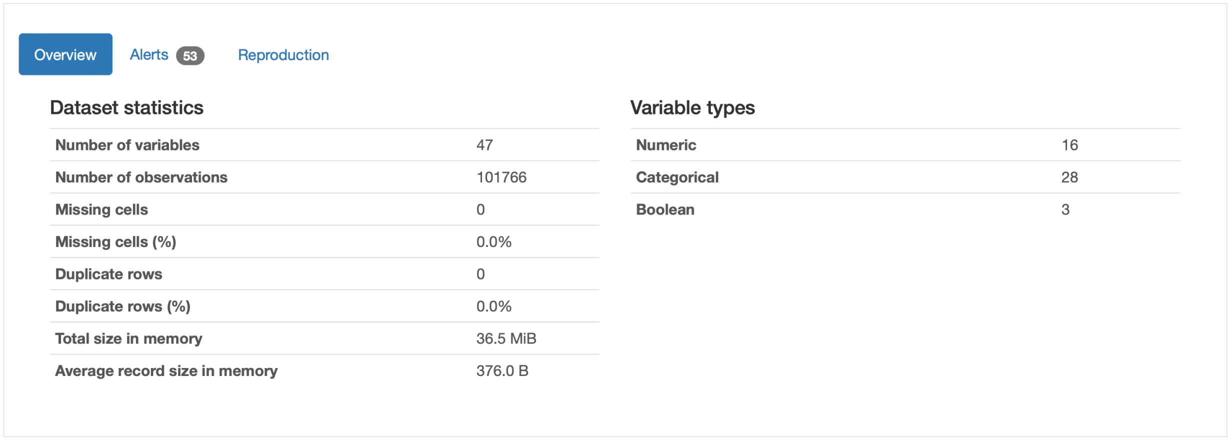
In [29]:

```
#diab_profile.to_widgets()
```

EDA Output

EDA Output from Profile Package Detailing the Outline of the data

Overview



Dataset statistics

Number of variables	47
Number of observations	101766
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	36.5 MiB
Average record size in memory	376.0 B

Variable types

Numeric	16
Categorical	28
Boolean	3

The above image details the outline of the data showing the various data types in the data. The data has 101766 rows and 50 columns of originally but 3 columns were dropped due significant missing values being more that our adpated appraoch of dropping 5% or more missing values. There are 47 columns left for further analysis of which 28 are categorical, 3 being boolean (true or false) and 16 are numerical.

EDA Output Cont

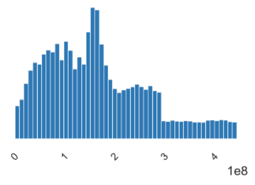
EDA Output from the Profile Package Detailing the Outline of the variables in the data

Variables

encounter_id
Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
UNIQUE

Distinct	101766	Minimum	12522
Distinct (%)	100.0%	Maximum	443867222
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	165201645.6	Memory size	795.2 KiB

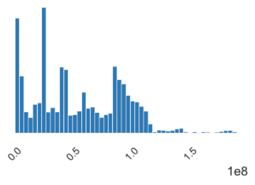


Toggle details

patient_nbr
Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	71518	Minimum	135
Distinct (%)	70.3%	Maximum	189502619
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	54330400.69	Memory size	795.2 KiB

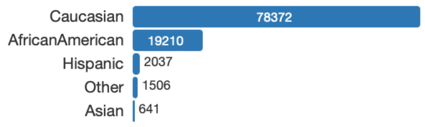


Toggle details

race
Categorical

HIGH CORRELATION

Distinct	5
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	795.2 KiB

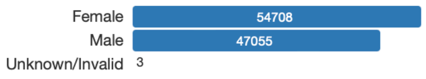


Toggle details

gender
Categorical

HIGH CORRELATION

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	795.2 KiB

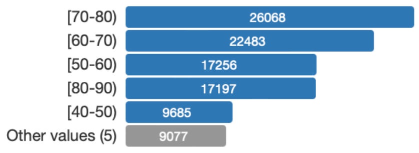


Toggle details

age
Categorical

HIGH CORRELATION

Distinct	10
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	795.2 KiB

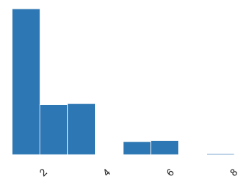


Toggle details

admission_type_id
Real number ($\mathbb{R}_{>0}$)

HIGH CORRELATION

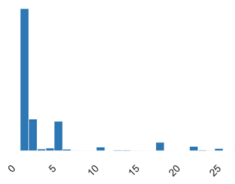
Distinct	8	Minimum	1
Distinct (%)	< 0.1%	Maximum	8
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	2.024006053	Memory size	795.2 KiB



Toggle details

discharge_disposition_id
Real number ($\mathbb{R}_{>0}$)

Distinct	26	Minimum	1
Distinct (%)	< 0.1%	Maximum	28
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	3.715641766	Memory size	795.2 KiB

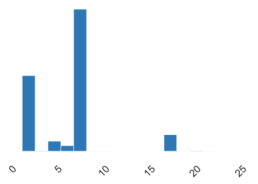


admission_source_id

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	17	**Minimum**	1
Distinct (%)	< 0.1%	**Maximum**	25
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	5.754436649	**Memory size**	795.2 KiB

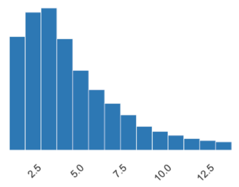


Toggle details

time_in_hospital

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	14	**Minimum**	1
Distinct (%)	< 0.1%	**Maximum**	14
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	4.395986872	**Memory size**	795.2 KiB

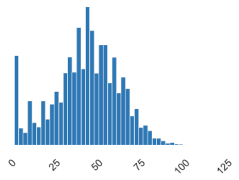


Toggle details

num_lab_procedures

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	118	**Minimum**	1
Distinct (%)	0.1%	**Maximum**	132
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	43.09564098	**Memory size**	795.2 KiB



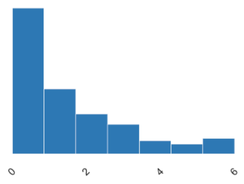
Toggle details

num_procedures

Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

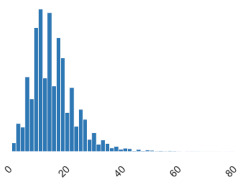
Distinct	7	**Minimum**	0
Distinct (%)	< 0.1%	**Maximum**	6
Missing	0	**Zeros**	46652
Missing (%)	0.0%	**Zeros (%)**	45.8%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	1.339730362	**Memory size**	795.2 KiB



Toggle details

num_medications
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	75	**Minimum**	1
Distinct (%)	0.1%	**Maximum**	81
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	16.02184423	**Memory size**	795.2 KiB

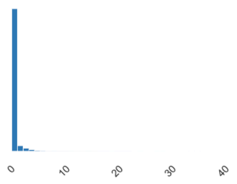


Toggle details

number_outpatient
Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Distinct	39	**Minimum**	0
Distinct (%)	< 0.1%	**Maximum**	42
Missing	0	**Zeros**	85027
Missing (%)	0.0%	**Zeros (%)**	83.6%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.3693571527	**Memory size**	795.2 KiB

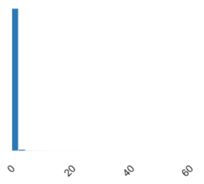


Toggle details

number_emergency
Real number ($\mathbb{R}_{\geq 0}$)

SKEWED
ZEROS

Distinct	33	**Minimum**	0
Distinct (%)	< 0.1%	**Maximum**	76
Missing	0	**Zeros**	90383
Missing (%)	0.0%	**Zeros (%)**	88.8%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.1978362125	**Memory size**	795.2 KiB

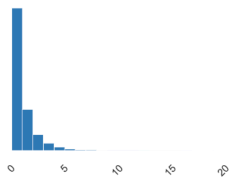


Toggle details

number_inpatient
Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Distinct	21	**Minimum**	0
Distinct (%)	< 0.1%	**Maximum**	21
Missing	0	**Zeros**	67630
Missing (%)	0.0%	**Zeros (%)**	66.5%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.6355659061	**Memory size**	795.2 KiB



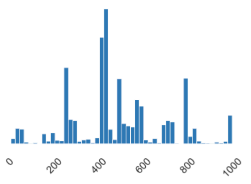
Toggle details

diag_1

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	709	**Minimum**	3
Distinct (%)	0.7%	**Maximum**	999
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	486.5183199	**Memory size**	795.2 KiB



Toggle details

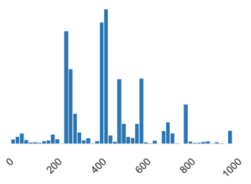
diag_2

Real number ($\mathbb{R}_{\geq 0}$)

diag_2

HIGH CORRELATION

Distinct	705	**Minimum**	2
Distinct (%)	0.7%	**Maximum**	999
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	434.4104617	**Memory size**	795.2 KiB

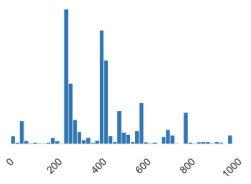


Toggle details

diag_3

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	726	**Minimum**	1
Distinct (%)	0.7%	**Maximum**	999
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	407.4365624	**Memory size**	795.2 KiB

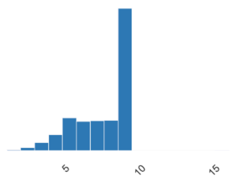


Toggle details

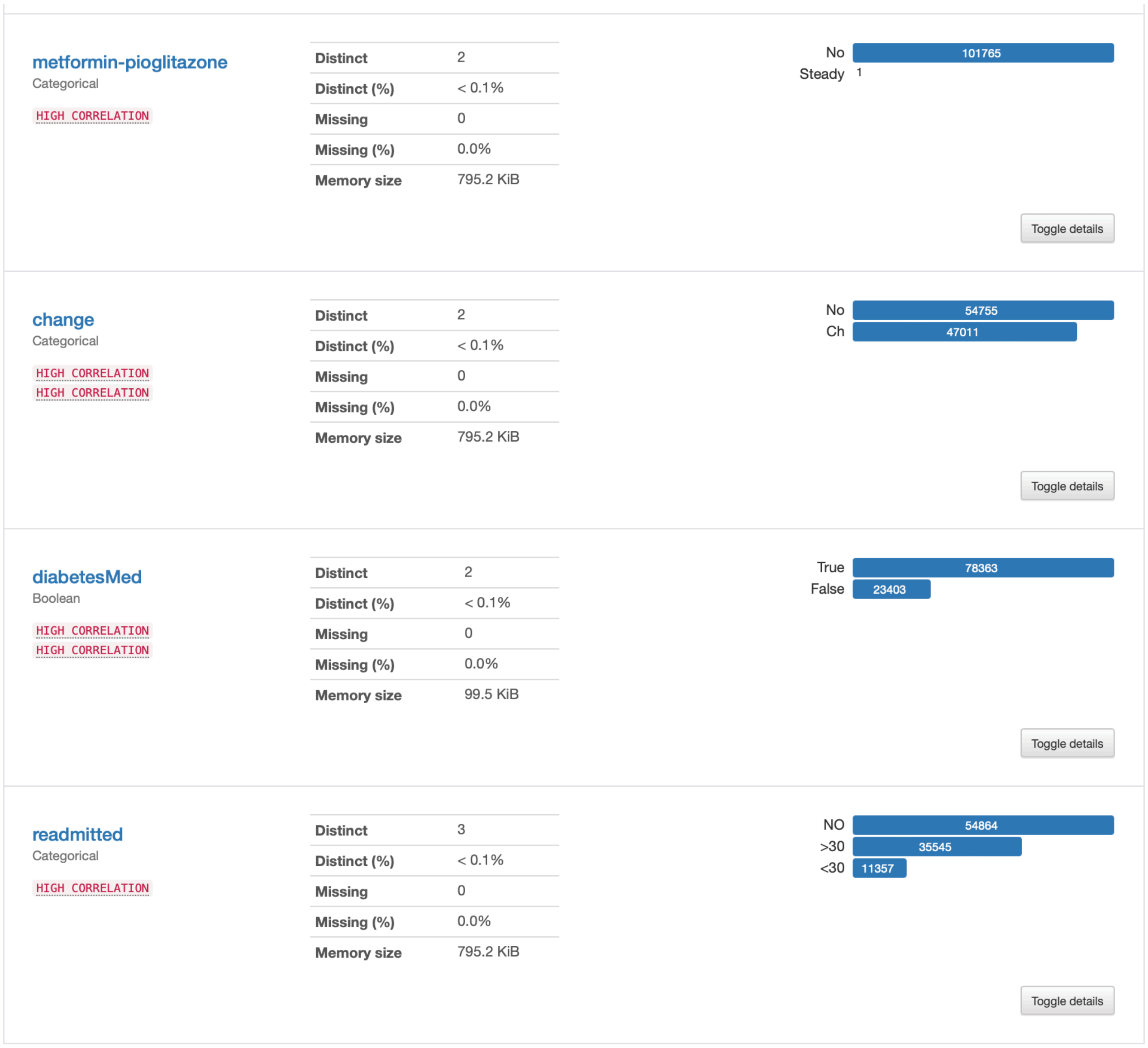
number_diagnoses

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	16	**Minimum**	1
Distinct (%)	< 0.1%	**Maximum**	16
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	7.422606765	**Memory size**	795.2 KiB



Toggle details



metformin-pioglitazone
Categorical
HIGH CORRELATION

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	795.2 KiB

No	101765
Steady	1

change
Categorical
HIGH CORRELATION
HIGH CORRELATION

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	795.2 KiB

No	54755
Ch	47011

diabetesMed
Boolean
HIGH CORRELATION
HIGH CORRELATION

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	99.5 KiB

True	78363
False	23403

readmitted
Categorical
HIGH CORRELATION

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	795.2 KiB

NO	54864
>30	35545
<30	11357

The outline shows that the data is heavily imbalanced (skewed data). These imply that data needs to be scaled by standardization by centering the data and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one for model interpretability.

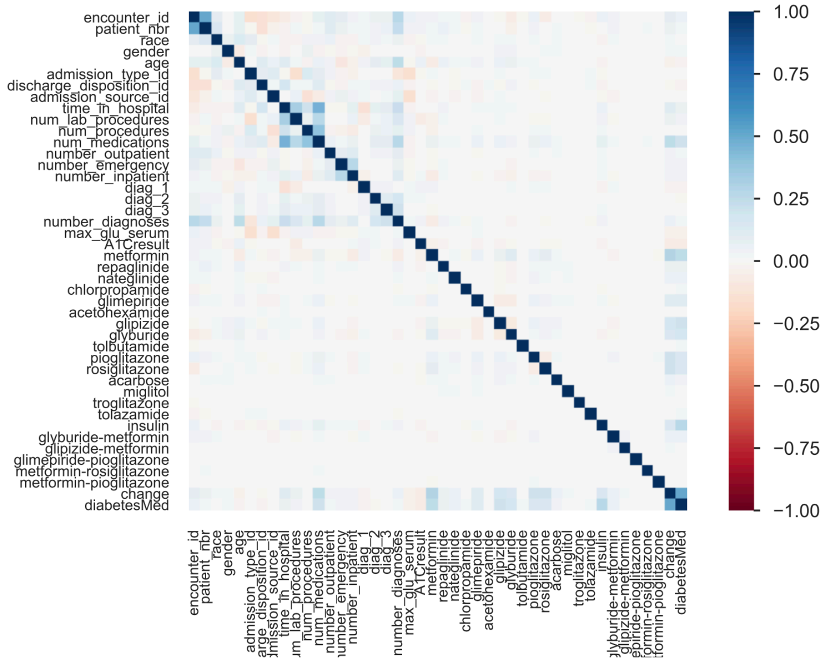
Source: https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

Feature Analysis using Correlation Matrix

Output from the Profile Package Detailing the Correlation Variables with Significance

Correlations

Spearman's ρ | Pearson's r | Kendall's τ | Cramér's V (φc) | Phik (φk)

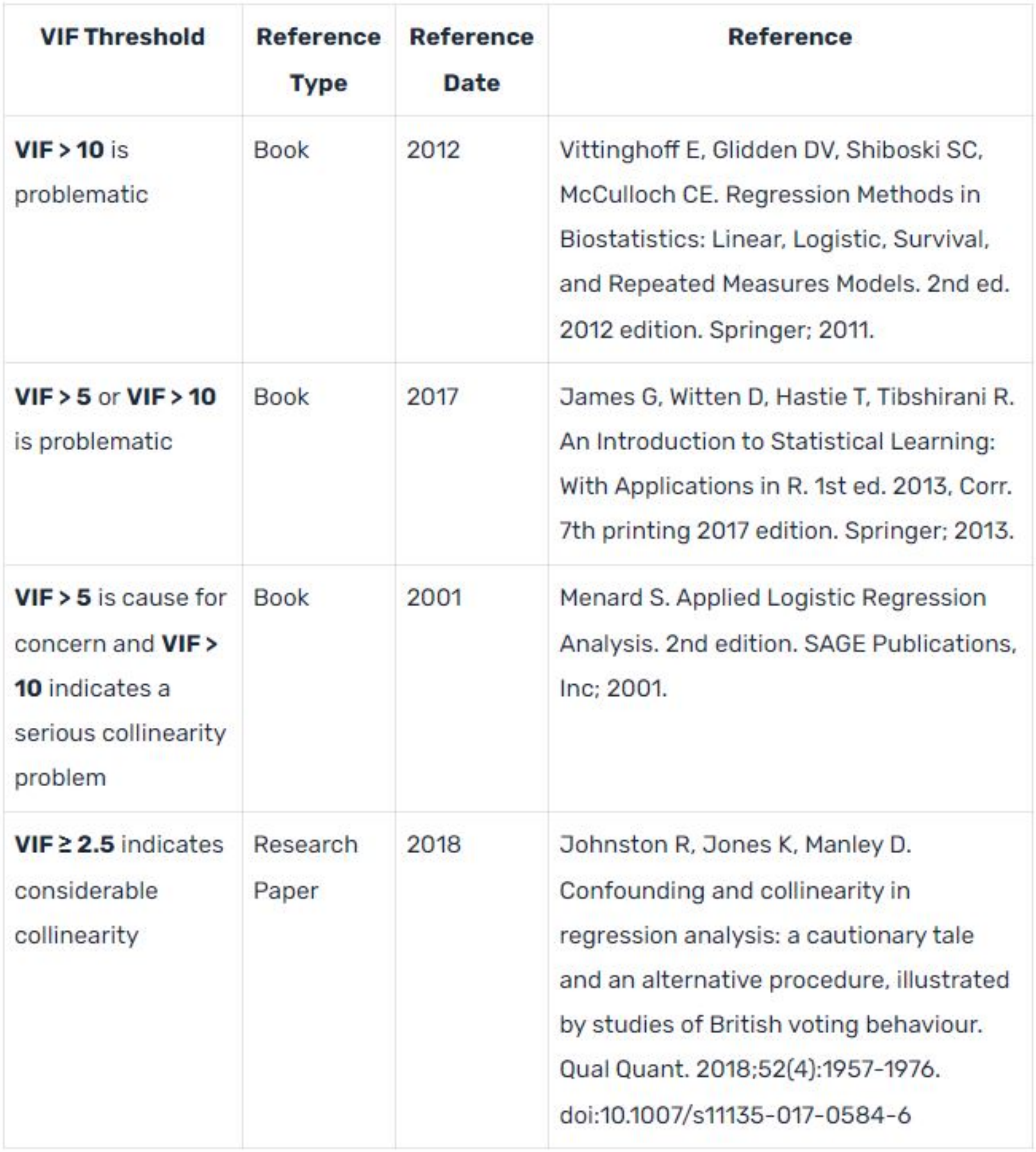


The above correlation matrix show some of the strong correlations between different features. As expected patient ID and patient number are correlated. time_spent_in_hospital is positively correlated to patients_time in_lab and number_of_medication . Also, number_of_medication is positively correlated to number_of_procedure. number_of_emergency_visits is positively correlated to number_of_impatient.

The Profile package was used to select features of relevance based on their correlation, using the Pearson's Correlation Metrics. We have gathered a list of important features for our model building using 21 out of 47 features but this will be probed further to firm up the decission to move forward with this assertion.

Although correlation matrix above can be a useful tool to find multicollinearity, their outcome only shows a bivariate relationship between the independent variables in our dataset, hence we are exploring other mean to check and deal with the problem of multicollinearity. A simple method to detect multicollinearity in a model is to use a Variance Inflation Factor(VIF) approach to get a better understanding at this.

The table below will be our reference point to tackle this problem. From the table, we will drop any VIF above 5 and to be more conservative use all VIF below 2.5 score.



VIF Threshold	Reference Type	Reference Date	Reference
VIF > 10 is problematic	Book	2012	Vittinghoff E, Glidden DV, Shiboski SC, McCulloch CE. Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models. 2nd ed. 2012 edition. Springer; 2011.
VIF > 5 or **VIF > 10** is problematic	Book	2017	James G, Witten D, Hastie T, Tibshirani R. An Introduction to Statistical Learning: With Applications in R. 1st ed. 2013, Corr. 7th printing 2017 edition. Springer; 2013.
VIF > 5 is cause for concern and **VIF > 10** indicates a serious collinearity problem	Book	2001	Menard S. Applied Logistic Regression Analysis. 2nd edition. SAGE Publications, Inc; 2001.
VIF ≥ 2.5 indicates considerable collinearity	Research Paper	2018	Johnston R, Jones K, Manley D. Confounding and collinearity in regression analysis: a cautionary tale and an alternative procedure, illustrated by studies of British voting behaviour. Qual Quant. 2018;52(4):1957-1976. doi:10.1007/s11135-017-0584-6

Source:https://quantifyinghealth.com/vif-threshold/

Feature Analysis Checking for Multicollinearity using VIF

Based on the VIF ouput in checking or detecting multicollinearity and dealing with its effect on our model, we will drop predictors with high collinerity from the dataset based on the desired threshold of at most 2.5 score for modeling.

In [30]:

```
# Calculation multicollinearity for each predictor
X = new_diab
```

In [31]:

```
# Import library and setting parameters for VIF computation
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calculate_vif_(X, thresh=2.5):
    variables = list(range(X.shape[1]))
    dropped = True
    while dropped:
        dropped = False
        vif = [variance_inflation_factor(X.iloc[:, variables].values, ix)
               for ix in range(X.iloc[:, variables].shape[1])]

        maxloc = vif.index(max(vif))
        if max(vif) > thresh:
            print('dropping \'' + X.iloc[:, variables].columns[maxloc] +
                  '\' at index: ' + str(maxloc))
            del variables[maxloc]
            dropped = True


    print('Remaining variables:')
    print(X.columns[variables])
    print(X.iloc[:, variables].shape[1])
```

In [32]:

```
calculate_vif_(X, thresh=2.5)
```

```
dropping 'glimepiride-pioglitazone' at index: 39
dropping 'miglitol' at index: 33
dropping 'chlorpropamide' at index: 24
dropping 'acarbose' at index: 31
dropping 'glyburide-metformin' at index: 34
dropping 'nateglinide' at index: 23
dropping 'repaglinide' at index: 22
dropping 'max_glu_serum' at index: 19
dropping 'number_diagnoses' at index: 18
dropping 'glimepiride' at index: 20
dropping 'rosiglitazone' at index: 25
dropping 'age' at index: 4
dropping 'pioglitazone' at index: 23
dropping 'glyburide' at index: 21
dropping 'glipizide' at index: 20
dropping 'metformin' at index: 18
dropping 'num_medications' at index: 10
dropping 'A1Cresult' at index: 16
dropping 'num_lab_procedures' at index: 8
dropping 'diabetesMed' at index: 24
dropping 'diag_2' at index: 13
dropping 'diag_1' at index: 12
dropping 'diag_3' at index: 12
dropping 'race' at index: 2
dropping 'insulin' at index: 15
dropping 'time_in_hospital' at index: 6
dropping 'admission_type_id' at index: 3
Remaining variables:
Index(['encounter_id', 'patient_nbr', 'gender', 'discharge_disposition_id',
       'admission_source_id', 'num_procedures', 'number_outpatient',
       'number_emergency', 'number_inpatient', 'acetohexamide', 'tolbutamide',
       'troglitazone', 'tolazamide', 'glipizide-metformin',
       'metformin-rosiglitazone', 'metformin-pioglitazone', 'change',
       'readmitted_binary'],
      dtype='object')
18
```

In [33]:

```
#Index of usable features
usable_ind_vars =['encounter_id', 'patient_nbr', 'gender', 'discharge_disposi
       'admission_source_id', 'num_procedures', 'number_outpatient',
       'number_emergency', 'number_inpatient', 'acetohexamide', 'tolbutamide'
       'troglitazone', 'tolazamide', 'glipizide-metformin',
       'metformin-rosiglitazone', 'metformin-pioglitazone', 'change',
       'readmitted_binary']
```

In [34]:

```
#dropping unwanted features based on their VIFs criteria defined above
model_diab_df = new_diab.drop(['glimepiride-pioglitazone', 'miglitol', 'chlor
                             'repaglinide','max_glu_serum','number_diagnoses','gl
                             'glipizide','metformin','num_medications','A1Cresult
                             'race','insulin','time_in_hospital','admission_type_
```

After VIF computations, these features were dropped based on thier high collinerity with other predictors in the dataset. These features are: 'glimepiride-pioglitazone', 'miglitol', 'chlorpropamide','acarbose','glyburide-metformin','nateglinide', 'repaglinide','max_glu_serum','number_diagnoses','glimepiride','rosiglitazone','age','pioglitazone', 'glipizide','metformin','num_medications','A1Cresult','num_lab_procedures','diabetesMed','diag_2 'race','insulin','time_in_hospital' and 'admission_type_id'.

This gives us a refined dataset to work with in building our model of which a total of 17 out of 49 features from the original dataset will be used after data cleaning and preparation while the target data remains 'readmitted_binary'. The features are 'encounter_id', 'patient_nbr', 'gender', 'discharge_disposition_id','admission_source_id', 'num_procedures', 'number_outpatient','number_emergency', 'number_inpatient', 'acetohexamide', 'tolbutamide', 'troglitazone', 'tolazamide', 'glipizide-metformin','metformin-rosiglitazone', 'metformin-pioglitazone'and 'change'.

In [35]:
```
# Checking out our refined data set after data cleaning and preparation
model_diab_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 18 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   encounter_id              101766 non-null  int64
 1   patient_nbr               101766 non-null  int64
 2   gender                    101766 non-null  int64
 3   discharge_disposition_id  101766 non-null  int64
 4   admission_source_id       101766 non-null  int64
 5   num_procedures            101766 non-null  int64
 6   number_outpatient         101766 non-null  int64
 7   number_emergency          101766 non-null  int64
 8   number_inpatient          101766 non-null  int64
 9   acetohexamide             101766 non-null  int64
 10  tolbutamide               101766 non-null  int64
 11  troglitazone              101766 non-null  int64
 12  tolazamide                101766 non-null  int64
 13  glipizide-metformin       101766 non-null  int64
 14  metformin-rosiglitazone   101766 non-null  int64
 15  metformin-pioglitazone    101766 non-null  int64
 16  change                    101766 non-null  int64
 17  readmitted_binary         101766 non-null  int64
dtypes: int64(18)
memory usage: 14.0 MB
```

In [36]:
```
#saving new dataset for modelling
model_diab_df.to_csv('diab_model_df.csv', index = False)
```

Model Preparation

We intend to evaluate our logistic regression model using three methods - a 70/30 split with no cross validation, K-Fold, and ShuffleSplit techniques.

Our base model will be a 70/30 train-test split to measure the performance of accuracy, F1-score, Precision, Recall, and AUC metrics, and then compare the metrics against the K-Fold and the ShuffleSplit technique.

Given our objective of predicting hospital re-admittance, the chosen method will help us build an effective model. The best competing model will help us to predict our target(re-admittance) response using unseen data.

We will measure the output of each approach to see which one gives us the best value in terms of Accuracy, F1-score, Precision, Recall, AUC metrics to decide which model is better.

To begin this analysis, the dataset needs to be standardized to make sure the content and the format are internally consistent. We standardize data when features have wide differences between ranges. For example, when there are numerical data with different measures (such as weight, distance, etc). the process helps the model to internalize the data and train itself effectively.

Source: https://builtin.com/data-science/when-and-why-standardize-your-data

In [37]:
```
# reading in the cleaned data for modeling
model_df = pd.read_csv('diab_model_df.csv')
```

In [38]:
```
model_df.head()
```

Out[38]:

	encounter_id	patient_nbr	gender	discharge_disposition_id	admission_source_id	num_pro
0	2278392	8222157	0	25	1	
1	149190	55629189	0	1	7	
2	64410	86047875	0	1	7	
3	500364	82442376	1	1	7	
4	16680	42519267	1	1	7	

Model Building & Evaluation

Model Evaluation

In [40]:

```
# First approach is to scale or standardized the data to center the predictor
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

sc = StandardScaler()
X =  pd.DataFrame(sc.fit_transform(model_df.drop(['readmitted_binary'],axis =

y = model_df['readmitted_binary']

X.head()
```

Out[40]:

	0	1	2	3	4	5	6	7	
0	-1.587330	-1.191545	-0.927397	4.031022	-1.169873	-0.785398	-0.291461	-0.21262	-0.50
1	-1.608075	0.033564	-0.927397	-0.514312	0.306482	-0.785398	-0.291461	-0.21262	-0.50
2	-1.608901	0.819654	-0.927397	-0.514312	0.306482	2.145781	1.286748	-0.21262	0.28
3	-1.604653	0.726480	1.078031	-0.514312	0.306482	-0.199162	-0.291461	-0.21262	-0.50
4	-1.609366	-0.305227	1.078031	-0.514312	0.306482	-0.785398	-0.291461	-0.21262	-0.50

In [186…

```
# 70/30 train and test slpit on df
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ran
```

In [183…

```
# 70/30 model with no cross validation using Logistics Regression

classify = LogisticRegression(random_state=0).fit(X_train, y_train)
pred_result = classify.predict_proba(X)
pred_result_class = classify.predict(X)

score = classify.score(X_test, y_test)


print('Accuracy score without validation is : ', score)
```

```
Accuracy score without validation is :  0.8849328529315428
[-2.43819579 -1.95179934 -2.16800043 ... -2.44017292 -2.04895162
 -2.48000782]
```

In [178…

```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection

predictions = LogisticRegression(random_state=123456).fit(X_test,y_test)
predict_result = predictions.predict_proba(X_test)
predict_result_class = predictions.predict(X_test)



#use model to predict probability that given y value is 1
y_pred = predictions.predict_proba(X_test)[::,1]

#calculate AUC of model
auc = metrics.roc_auc_score(y_test, y_pred)

#print AUC score
print('This is AUC:', auc)

print(classification_report(y_test,predict_result_class))
```

```
This is AUC: 0.6342195527283687
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     27035
           1       0.48      0.02      0.03      3495

    accuracy                           0.89     30530
   macro avg       0.69      0.51      0.49     30530
weighted avg       0.84      0.89      0.84     30530
```

In [179…

```
#Implementing cross validation (cross_val_score) using Logistics Regression
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn import datasets, linear_model
from sklearn.model_selection import cross_val_score
import sklearn.metrics as metrics

model = LogisticRegression(random_state = 123456).fit(X_train, y_train)

y_pred = cross_val_predict(model, X_train, y_train, cv=10)

scores = cross_val_score(model, X_train, y_train, cv=5)

f1_score = cross_val_score(model, X_train, y_train, cv=5,scoring='f1_macro')

precision = cross_val_score(model, X_train, y_train, cv=5,scoring='precision'

recall = cross_val_score(model, X_train, y_train, cv=5,scoring='recall')

roc_auc = cross_val_score(model, X_train, y_train, cv=5,scoring= 'roc_auc')

print('This is Accuracy:',scores.mean())

print('This is F1_Score:',f1_score.mean())

print('This is Precision:',precision.mean())

print('This is Recall:',recall.mean())

print('This is AUC:',roc_auc.mean())
```

```
This is Accuracy: 0.8894659905941957
This is F1_Score: 0.48584354923615536
This is Precision: 0.479160310588882
This is Recall: 0.015644568246927772
This is AUC: 0.6290506234805731
```

In [194…

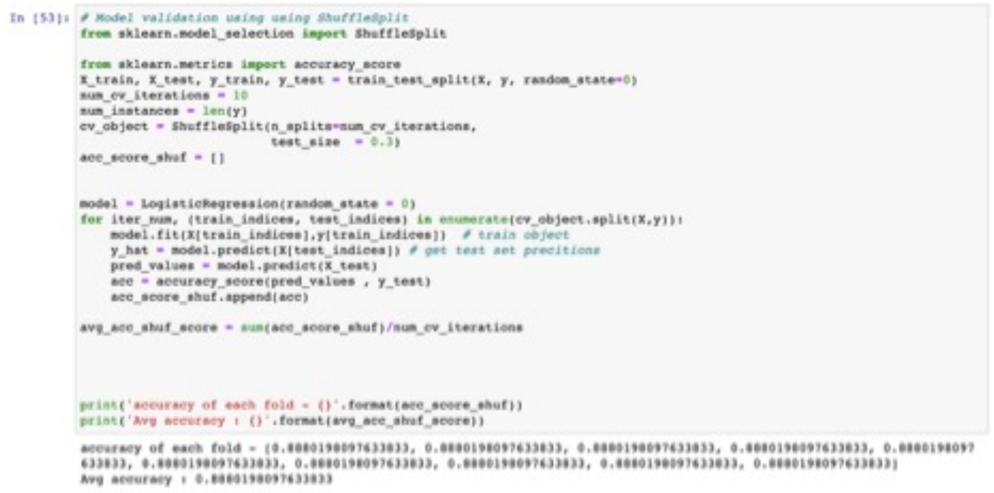
```
# Model validation using using ShuffleSplit
#from sklearn.model_selection import ShuffleSplit
#from sklearn.linear_model import LogisticRegression
#from sklearn.metrics import accuracy_score
#X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
#num_cv_iterations = 10
#num_instances = len(y)
#cv_object = ShuffleSplit(n_splits=num_cv_iterations,
#                          test_size  = 0.3)
#acc_score_shuf = []


#model = LogisticRegression(random_state = 1)
#for iter_num, (train_indices, test_indices) in enumerate(cv_object.split(X,y
#    model.fit(X[train_indices],y[train_indices])  # train object
#    y_hat = model.predict(X[test_indices]) # get test set precitions
#    pred_values = model.predict(X_test)
#    acc = accuracy_score(pred_values , y_test)
#    acc_score_shuf.append(acc)

#avg_acc_shuf_score = sum(acc_score_shuf)/num_cv_iterations




#print('accuracy of each fold - {}'.format(acc_score_shuf))
#print('Avg accuracy : {}'.format(avg_acc_shuf_score))
```



In [53]:

```
# Model validation using using ShuffleSplit
from sklearn.model_selection import ShuffleSplit

from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
num_cv_iterations = 10
num_instances = len(y)
cv_object = ShuffleSplit(n_splits=num_cv_iterations,
                         test_size  = 0.3)
acc_score_shuf = []


model = LogisticRegression(random_state = 0)
for iter_num, (train_indices, test_indices) in enumerate(cv_object.split(X,y)):
    model.fit(X[train_indices],y[train_indices])  # train object
    y_hat = model.predict(X[test_indices]) # get test set precitions
    pred_values = model.predict(X_test)
    acc = accuracy_score(pred_values , y_test)
    acc_score_shuf.append(acc)

avg_acc_shuf_score = sum(acc_score_shuf)/num_cv_iterations




print('accuracy of each fold - {}'.format(acc_score_shuf))
print('Avg accuracy : {}'.format(avg_acc_shuf_score))
```

```
accuracy of each fold - [0.8880198097633833, 0.8880198097633833, 0.8880198097633833, 0.8880198097633833, 0.8880198097
633833, 0.8880198097633833, 0.8880198097633833, 0.8880198097633833, 0.8880198097633833, 0.8880198097633833]
Avg accuracy : 0.8880198097633833
```

NB: when running this code, it produces an error on this work computer but runs on another computer.

In [105…

```
#from sklearn.neighbors import KNeighborsRegressor

#knn_model = KNeighborsRegressor(n_neighbors=3)

#knn_model.fit(X_train, y_train)

from sklearn.neighbors import KNeighborsClassifier
import sklearn.metrics as metrics
# Create KNN classifier
model = LogisticRegression(random_state = 2)
knn = KNeighborsClassifier(n_neighbors = 8)
# Fit the classifier to the data
knn.fit(X_train,y_train)

y_pred = knn.predict(X_test)

print(metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
0.8841794955781199
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     27035
           1       0.38      0.02      0.03      3495

    accuracy                           0.88     30530
   macro avg       0.63      0.51      0.49     30530
weighted avg       0.83      0.88      0.83     30530
```

In [134…

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import make_scorer
from sklearn.metrics import confusion_matrix

model = LogisticRegression(random_state = 123456)
def confusion_matrix_scorer(model, X, y):
    y_pred = model.predict(X_train)
    cm = confusion_matrix(y_train, y_test)
    return {'tn': cm[0, 0], 'fp': cm[0, 1],
            'fn': cm[1, 0], 'tp': cm[1, 1]}
#cv_results = cross_validate(model, X_train, y_train, cv=5,scoring= 'f1_macro
scores = cross_val_score(model, X_train, y_train, cv=5, scoring='f1_macro')
scores
```

Out[134…

```
array([0.48993069, 0.48750073, 0.48536558, 0.48210634, 0.4843144 ])
```

In [136…

```
from sklearn.model_selection import ShuffleSplit
n_samples = X.shape[0]
cv = ShuffleSplit(n_splits=5, test_size=0.3, random_state=0)
cross_val_score(model, X, y, cv=cv)
```

Out[136…

```
array([0.88765149, 0.88955126, 0.8887979 , 0.89092696, 0.88683262])
```

In [87]:

```
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import BernoulliNB
#import sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
nb = BernoulliNB()
param_grid = {'alpha':[1000,100,10,1,0.1,0.01,0.001]} #params we need to try
gsv = GridSearchCV(nb,param_grid,cv=2,verbose=1,n_jobs=-1,scoring='f1')
gsv.fit(X_train,y_train)
nb = BernoulliNB(alpha=0.1)
nb.fit(X_train,y_train)
train_pred = nb.predict(X_train)
cv_pred = nb.predict(X_train)
test_pred = nb.predict(X_test)
y_prob = nb.predict_proba(X_train)
print("Train Set Accuracy: {}".format(metrics.accuracy_score(train_pred, y_tr
```

```
Fitting 2 folds for each of 7 candidates, totalling 14 fits
Train Set Accuracy: 0.8896344544893031
```

For our logistic regression model, we have applied three different methods to validate the model: 70/30 split, KNN, and shuffleSplit. 70/30 split method splits the dataframe into 70% train and 30% test. After we predicted using our test features using this cross-validation method, we were able to produce an accuracy rate of 88.79%. Precision is 89%, recall is 100%, f1 is 94% for readmittance, non-readmittance precision 48%, recall 2%, f1 3%

KNN method used to predict on test data using the K using the trained data from nearest Neighbor value (K value). Instead of splitting the data into two parts, data gets split into K parts – 8 parts. This validation process produces accuracy rate of 88.4%. Precision is 89%, recall is 100%, f1 is 94%. For re-admittance, non-readmittance precision 38%, recall 2%, f1 3%

ShuffleSplit method used random samples from the entire dataset, random test and train sets are created during 10 iteration process. The method produces a final accuracy score using the average of 10 accuracy scores from the iteration run. This model produced accuracy score of 88.80%

Given the performance of three of the above methods, we prefer ShuffleSplit method for training our model. This method has produced the top scores for our performance metrics, for example, it has produced the top accuracy score of 88.80%, which is slightly above of 70/30 split method of 88.79%

Model Interpretability and Explainability

The following are the top ten important features for our model:

Encounter_id Patient_nbr gender Discharge_disposition Admission_source_id Num_procedures Num_medications Number_outpatient Number_emergency Number_impatient

These variables are important based on the higher coefficient values of those features. We find these features to be more accurate in predicting the target variable of hospital readmission.

Conclusion

We notice that cross-validation improves the performance of the model by reducing overfitting. We recommend ShuffleSplit cross-validation method since it has the best performance metrics although we have noticed 70/30 split method's scores are close to that method. In terms of patient demographics, gender is one of the top features out of 10 overall important features of patient re-admittance. Patient visits during both regular and emergency, number of procedures, number of medication consumed by the patients are very important features that hospital may find useful for their operation strategies.

Recommendation

In addition, we noticed that when recall rate increases, this has an inverse effect on preccion and accuracy rate. Therefore, the team recommends that management reduce the recall addmitance rate to improve on accuracy, and precision rate. This will save money and resources to cater to new arrivals.

```
In [ ]:
```