CAPSTONE PROJECT GROUP 5

Predicting Diabetic Patients Readmission within 30 Days

Interim Report

Mentored By:
Mr. Sravan Malla

Submitted By:
Ms. Anita H
Mr. Aroop Ajaykumar
Ms. Mrunal Dalvi
Mr. Rushikesh Hanjankar
Ms. Shweta Chavan

Table of Content

Abstract

The healthcare domain faces a new a challenge that will affect its entire operation and execution. As healthcare organizations shift focus to high quality and care, managing cost has been challenging. A large portion of TCOC (Total Cost of Care) comes from patients who make multiple unscheduled hospital visits for the same underlying pathology: a hospital readmission. We can use predictive modeling from Machine learning to help prioritize patients. One of the patient population that is at an increased risk of hospitalization and readmission is that of diabetes. Diabetes is a medical condition that affects approximately 1 in 10 patients in the world. Here, in this project we aim to implement various tools on features and compare the performance of multiple classifiers to predict the hospital readmission for diabetic patients.

Introduction

Diabetes is a very common metabolic disease and is a growing burden in the world. Chronic complications of Diabetes include retinopathy, neuropathy, nephropathy, an increased risk of cardiovascular disease and major cardiac events including myocardial infarction and stroke. The high prevalence of Diabetes and its complications makes it a common condition in hospitalized patients. This leads to frequent admissions for procedures and interventions during which patients with Diabetes are reported to have longer lengths of stay, increased hospital complications, and mortality.

In recent years, government agencies and healthcare systems are increasingly focused on 30-day readmission rates to improve quality and determine the complexity of patient's populations. The centers for Medicare and Medicaid services (CMS) has labeled 30-day readmission rates as a measure of healthcare quality and emphasize its reduction as a strategy to reduce healthcare cost while also maintaining quality. As the healthcare system moves toward value-based care, CMS has created many programs to improve the quality of care of patients. One of these programs is called the Hospital Readmission Reduction Program (HRRP), which reduces reimbursement to hospitals with above average readmissions. For those hospitals which are currently penalized under this program, one solution is to create interventions to provide additional assistance to patients with increased risk of readmission. But how do we identify these patients? We can use predictive modeling from data science to help prioritize patients. Our research will focus on finding the high-risk patients who are prone to coming back and getting

readmitted for less than 30-day readmission. The diabetic patient population offers a large data set, and a large opportunity for healthcare cost savings with proper care. The average medical spending for diabetes patients is 2.3 times more than similar patients without diabetes.

Problem Statement

Predict if a patient with diabetes will be readmitted to the hospital within 30 days. As, if there is a patient getting readmitted within 30 days, the hospital faces incentive issues and also increases the financial expenses of the patient. Taking this into consideration, the aim is to correctly predict the readmission of a diabetic patient.

Literature Review

The data in this project is a classic example of Supervised Machine Learning, Classification Problem. In analytics, the classification problems are an important category where the outcome variable takes discrete variables. Its primary objective is Class Probability i.e. to predict the probability of an observation belonging to a particular class. Classification problems are generally of two types, namely Binary and Multinomial Classification. In this data, there are target observations belonging to different classes, thus is a multinomial classification. However, considering the problem statement and the business goal we convert it to a binary classification to further aid in correct model building.

As it is now a Binary Classification, we use Logistic Regression as base model. Then we move on to more complex models like Decision Tree, Random Forest, Bagging & Boosting along with Hyperparameter Tuning and further selecting the best from those depending on F1-score, accuracy, confusion matrix and other evaluation matrices.

Data Description

The dataset contains the diabetic patient information who was admitted to a particular hospital. The data is of various patients who have been readmitted into the hospital numerously. Dataset comprises of 101766 rows and 50 columns. The data also consists of 13 integers and 37 object variables. The most important column we can see is readmitted, it tells us if a patient was readmitted to the hospital within 30 days, greater than 30 days or not readmitted. The data recorded over a span of 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks.

Dataset Dictionary

List of features and their descriptions in the initial dataset (the dataset is also available at the website of Data Mining and Biomedical Informatics Lab at VCU (http://www.cioslab.vcu.edu/)).

Sr.	Feature	Description
1	Encounter ID	Unique identifier of an encounter
2	Patient number	Unique identifier of a patient
3	Race	Values: Caucasian, Asian, African American, Hispanic, and other
4	Gender	Values: male, female, and unknown/invalid
5	Age	Grouped in 10-year intervals: 0, 10), 10, 20), …, 90, 100)
6	Weight	Weight in pounds.
7	Admission type	Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available
8	Discharge disposition	Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available
9	Admission source	Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital
10	Time in hospital	Integer number of days between admission and discharge
11	Payer code	Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay
12	Medical specialty	Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon
13	Number of lab procedures	Number of lab tests performed during the encounter
14	Number of procedures	Number of procedures (other than lab tests) performed during the encounter
15	Number of medications	Number of distinct generic names administered during the encounter
16	Number of outpatient visits	Number of outpatient visits of the patient in the year preceding the encounter
17	Number of emergency visits	Number of emergency visits of the patient in the year preceding the encounter
18	Number of inpatient visits	Number of inpatient visits of the patient in the year preceding the encounter
19	Diagnosis 1	The primary diagnosis (coded as first three digits of ICD9); 848 distinct values
20	Diagnosis 2	Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
21	Diagnosis 3	Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values
22	Number of diagnoses	Number of diagnoses entered to the system
23	Glucose serum test result	Indicates the range of the result or if the test was not taken. Values: “>200,” “>300,” “normal,” and “none” if not measured
24	A1c test result	Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>7” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured.
25	Change of medications	Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”
26	Diabetes medications	Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”
27	24 features for medications	For the generic names: metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone,

		metformin-rosiglitazone, and metformin-pioglitazone, the feature indicates whether the drug was prescribed or there was a change in the dosage. Values: “up” if the dosage was increased during the encounter, “down” if the dosage was decreased, “steady” if the dosage did not change, and “no” if the drug was not prescribed
28	Readmitted	Days to inpatient readmission. Values: “<30” if the patient was readmitted in less than 30 days, “>30” if the patient was readmitted in more than 30 days, and “No” for no record of readmission.
29	Admission_id	1 Emergency 2 Urgent 3 Elective 4 Newborn 5 Not Available 6 NULLS 7 Trauma Center 8 Not Mapped

Observations and challenges:

- In our dataset we have the 13 integer & 47 object variables.
- Patient nbr and encounter_id are identifiers. Hence, we will remove these columns.
- Weight and age variables are categorical. They are given within the range of 10 intervals.
- admission_type_id, discharge_disposition_id, admission_source_id are of integer datatypes, but they are label encoded and in the ID_mapping csv file we have their description. They are given as an identifier.
- Columns examide & citoglipton have exactly one unique value, hence we will remove these columns.
- diag1, diag2, diag3 represent the first 3 digits of the ICD9. They are numeric representation of the names of the medical diagnosis’s tests. There are more than 700 unique values in these variables.
- (for more information refer below link: https://en.wikipedia.org/wiki/List_of_ICD-9_codes)
- Null values:
- There are lot of null value in our dataset. Columns weight, payer code and medical_speciality have the highest percentage of null values.

- We will drop weight as that column has high number of null values; the payer code is mode of payment and has 46% missing values, assuming it will not affect the readmission we will drop it and check its effect.
- medical_speciality — It has 49% missing values (Medical specialty of the attendant surgeon), so we should consider this when making features. There are 2 ways we may either delete the whole column or impute with "unknown" value.
- For the rest missing values of the features diag_1, diag_2, diag_3 the percentage is very low we can delete missing values in these rows.
- Since we are primarily interested in factors that lead to early readmission, we defined their admission attribute(outcome) as having two values: impute ">30", "0" to "0" and ">30" to 1

Exploratory Data Analysis (EDA)

Dimensions:

Dataset comprises of 101766 rows and 50 columns. With 13 Integer datatype and 37 of object datatype.

```
print('Number of Rows:{}, Number of columns:{}'.format(df.shape[0],df.shape[1]))
print('='*80)
print('\nDatatypes Count\n',df.get_dtype_counts())

#print(df.select_dtypes(object).columns)
#print(df.select_dtypes('int64').columns)
```

```
Number of Rows:101766, Number of columns:50
================================================================================

Datatypes Count
 int64     13
object    37
dtype: int64
```

Univariate Analysis:

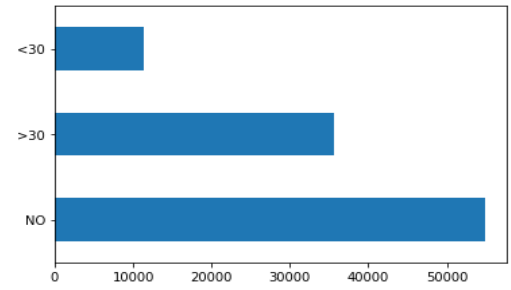
Target Variable:

Our target variable readmitted is very unbalanced, as we are going to impute "<30" and "NO" with "0". The unbalanced scale will increase exponentially in favor of "0".

```
display(np.round(df.readmitted.value_counts()/df.shape[0]*100))
df.readmitted.value_counts().plot(kind='barh')
```

```
NO     54.0
>30    35.0
<30    11.0
Name: readmitted, dtype: float64
```

```
0]: <matplotlib.axes._subplots.AxesSubplot at 0x1f6522e5a20>
```

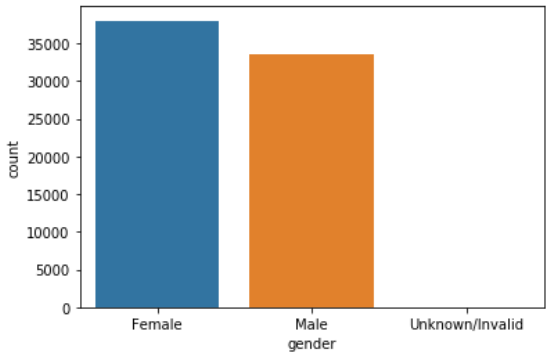


Gender:

As there are only 3 unknown or invalid gender type we can impute with "Female" as they are maximum in dataset.

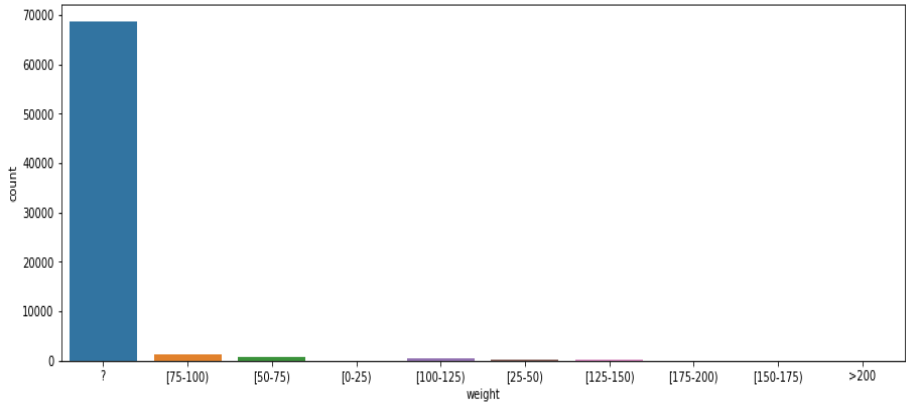
```
print(df.gender.value_counts())
sns.countplot(df.gender)
```

```
Female             38025
Male               33490
Unknown/Invalid        3
Name: gender, dtype: int64

<matplotlib.axes._subplots.AxesSubplot at 0x2894334ca90>
```



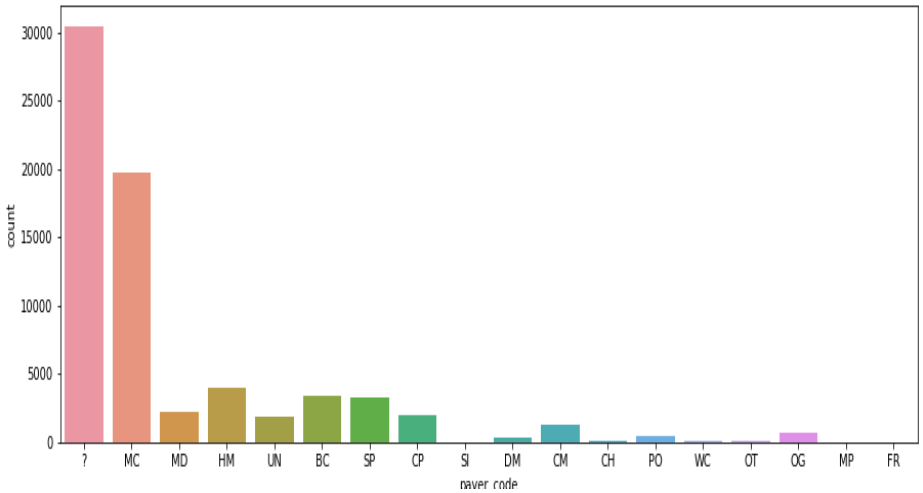
Weight:

There are 96% null values present in the weight column hence, we will delete this column.



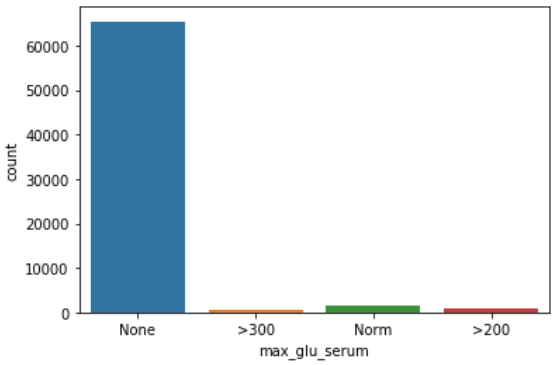
Payer_code:

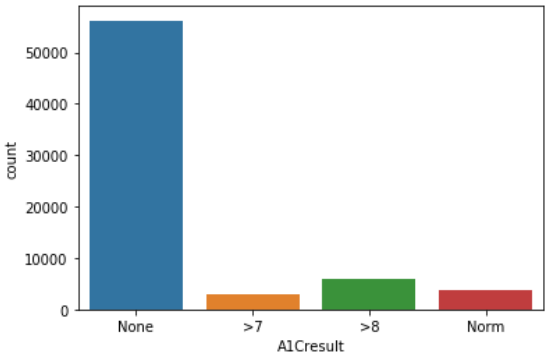
There are 46% null values and this column represent the mode of payment. Readmittance cannot be decided by the mode of payment. we can further use techniques and simplify the data depending upon their significance.



max_glu_serum and A1Cresult:

- The reference values for a "normal" random glucose test in an average adult are 80–140mg/dl (4.4–7.8 mmol/l), between 140-200mg/dl (7.8–11.1 mmol/l) is considered pre-diabetes, and ≥ 200 mg/dl is considered diabetes according to ADA guidelines.
- For people without diabetes, the normal range for the hemoglobin A1c level is between 4% and 5.6%. Hemoglobin A1c levels between 5.7% and 6.4% mean you have a higher chance of getting diabetes. Levels of 6.5% or higher mean you have diabetes





Missing Value Analysis:

Here we used a Function that will return the columns names as index, null_value_count, any unique character we specify & its percentage of occurrences per column. Overall null values, unique characters. There are lot of "?" value in our dataset. Columns weight, payer code and medical_speciality have the highest percentage of "?" values. We will drop weight as they have high number of null values; the payer code is mode of payment it will not affect the readmission hence we will drop it too. Medical specialty — It has 49% missing values (Medical specialty of the attendant surgeon), so we should consider this when making features. There are 2 ways we may either delete the whole column or impute with "unknown" value. For the rest missing values of the features diag_1, diag_2, diag_3 the percentage is very low we can delete missing values in these rows.

```
#Dataset Summary:
#%%time
def fun_view():
    |
    null_values = df.apply(lambda x:x.isnull().sum())
    blank_char = df.apply(lambda x:x.isin(['?']).sum())
    percent_blank_char = df.apply(lambda x:round((x.isin(['?']).sum()/df.shape[0])*100, 2))
    unique_values = df.apply(lambda x:len(x.unique()))
    return pd.DataFrame({'null_values':null_values,
                         '? Values':blank_char,'% ? Values':percent_blank_char
                        ,'unique_values':unique_values})
```

```
%%time
print('Function Information',fun_view.__doc__)
view = fun_view()
display(view)
```

```
Function Information  This Function will return the columns names as index,null_value_count,any unique character we specify
& its percentage of occurance per column.
```

```
# Overall null values in the dataset.
view[view['% ? Values'] != 0]
```

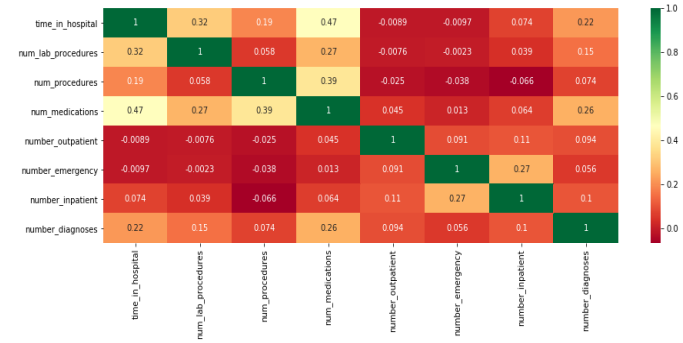
]:

	null_values	? Values	% ? Values	unique_values
race	0	2273	2.23	6
weight	0	98569	96.86	10
payer_code	0	40256	39.56	18
medical_specialty	0	49949	49.08	73
diag_1	0	21	0.02	717
diag_2	0	358	0.35	749
diag_3	0	1423	1.40	790

Correlation Analysis and Data distribution For Numeric Features:

```
plt.figure(figsize=(15,5))
sns.heatmap(df[numeric_var].corr(),cmap='RdYlGn',annot=True)
```

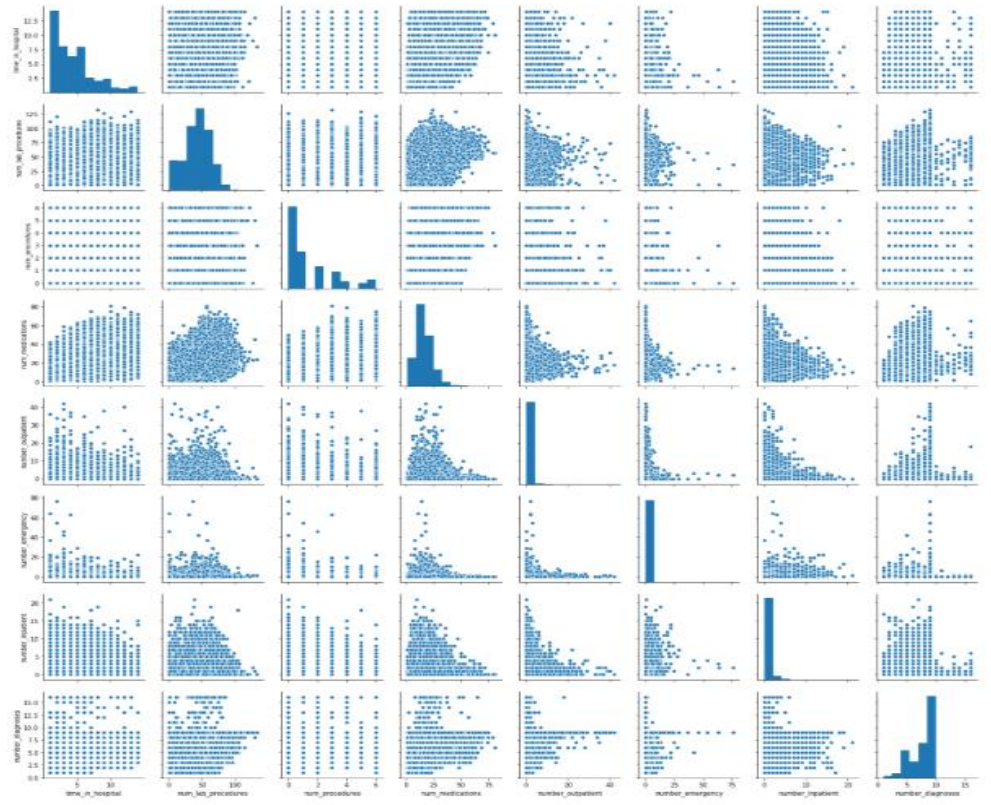
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f65a0922e8>
```



Pair plot:

```
numeric_var =  ['time_in_hospital','num_lab_procedures','num_procedures','num_medications','number_outpatient'
                ,'number_emergency','number_inpatient','number_diagnoses']

#sns.boxplot(df['num_medications'])
sns.pairplot(df[numeric_var])
```

```
<seaborn.axisgrid.PairGrid at 0x1f651e35470>
```



Outlier Detection:

Since some of the variables are skewed in nature, extreme values are present for couple of variables. We didn't do any outliers transformation There are outliers present with the target value 0 and 1. As the medical conditions of the patient varies, the number of medicines, number of diagnosis will also vary. As we don't have the full medical history of the patient, we will not remove the outliers as they might be genuine.

```
def outlier(temp1,nm):
    l = nm
    r=1.5
    Q3 = temp1[l].quantile(q=.75)
    Q1 = temp1[l].quantile(q=.25)
    IQR = Q3-Q1
    upper_limit = Q3 + (IQR*r)
    lower_limit = Q1 - (IQR*r)
    outlier_no = temp1[(temp1[l] < lower_limit) | (temp1[l] > upper_limit)].shape
    #withou_outlier =
    print('Out of {} observations, total outliers in feature {} is {}'.format(temp1.shape[0],l.upper(),outlier_no[0]))
    print('Mean without outlier:',temp1[(temp1[l] > lower_limit) & (temp1[l] < upper_limit)][l].mean())
    print('Mean with outlier',temp1[l].mean())
    plt.figure(figsize=(15,3))
    sns.boxplot(temp1[l])
```

Checking duplicates:

encounter_id patient_nbr:

There are no duplicate values present in the Encounter id. But in the patient_nbr feature 30248 duplicate values present. we will delete the id columns later during feature selection.

For example, patient number 23043240 have more than one encounter. Here we will keep only the first occurrence of the entry and delete the rest of the duplicates.

```
df.encounter_id.duplicated().sum()
```

```
0
```

```
df.patient_nbr.duplicated().sum()
```

```
30248
```

```
print('Before removing duplicate entries',df.shape)
df = df.drop_duplicates(subset= ['patient_nbr'], keep = 'first')
print('After removing duplicate entries',df.shape)
```

```
Before removing duplicate entries (101766, 50)
After removing duplicate entries (71518, 50)
```

Preprocessing and Data preparation:

Missing value imputation:

- we can see that there are "?" symbols present in the variable Weight, payer_code, medical_speciality. It seems due to unavailability of the information the "?" symbol was put in the records. We will replace this with nan values as the "?" in the column "race" will be replaced by the "Other"

```
df.replace('?',np.nan,inplace=True) # Replace Null values with Np.nan

df.race.fillna('Other',inplace=True) # replace null in Race with the 'Other'
```

Imputing the Diag_1, diag_2, diag_3:

Here we first drop the null values and then we dummify the ICD-9 codes to our columns as per International Statistical Classification of Diseases and Related Health Problems.

```
# Remove null values in diag_1, diag_2, diag_3examide
df.dropna(inplace=True)
df.shape
```

```
(97825, 43)
```

ICD-9 diagnosis codes contain three, four, or five digits. The three-digit code is the heading of a section of codes that are further divided by more detailed fourth and fifth digits.

we can see after dummifying the diag_1, diag_2, diag_3 there will lots of extra variables will be created to avoid that we must dummify only those with high number of occurrences.

```
# Creating the dictionary for ICD-9 imputation

dict11 = {'infectious and parasitic diseases':(1,139)
         ,'neoplasms':(140,239)
         ,'endocrine, nutritional and metabolic diseases, and immunity disorders':(240,279)
         ,'diseases of the blood and blood-forming organs':(280,289)
         ,'mental disorders':(290,319)
         ,'diseases of the nervous system and sense organs':(320,389)
         ,'diseases of the circulatory system':(390,459)
         ,'diseases of the respiratory system':(460,519)
         ,'diseases of the digestive system':(520,579)
         ,'diseases of the genitourinary system':(580,629)
         ,'complications of pregnancy, childbirth, and the puerperium':(630,679)
         ,'diseases of the skin and subcutaneous tissue':(680,709)
         ,'diseases of the musculoskeletal system and connective tissue':(710,739)
         ,'congenital anomalies':(740,759)
         ,'certain conditions originating in the perinatal period':(760,779)
         ,'symptoms, signs, and ill-defined conditions':(780,799)
         ,'injury and poisoning':(800,999)
         ,'external causes of injury and supplemental classification':(1000,2000)}
```

```
df[['diag_1','diag_2','diag_3']].head()
```

	diag_1	diag_2	diag_3
1	endocrine, nutritional and metabolic diseases,...	endocrine, nutritional and metabolic diseases,...	endocrine, nutritional and metabolic diseases,...
2	complications of pregnancy, childbirth, and th...	endocrine, nutritional and metabolic diseases,...	external causes of injury and supplemental cla...
3	infectious and parasitic diseases	endocrine, nutritional and metabolic diseases,...	diseases of the circulatory system
4	neoplasms	neoplasms	endocrine, nutritional and metabolic diseases,...
5	diseases of the circulatory system	diseases of the circulatory system	endocrine, nutritional and metabolic diseases,...

Age Imputation:

For the Age Group we may only consider the max value range. Like for [0-10] will be replaced by 10. & [56-60] by 60 we will use binning here.

```
# Age imputation:
age_imp = {'[0-10)':10,
           '[10-20)':20,
           '[20-30)':30,
           '[30-40)':40,
           '[40-50)':50,
           '[50-60)':60,
           '[60-70)':70,
           '[70-80)':80,
           '[80-90)':90,
           '[90-100)':100}

df.age = df.age.replace(age_imp)
```

IDs Imputation:

We will make a dictionary based on data in the ID_mapping CSV file to impute columns ad_type_id, discharge_disposition_id, admission_sid.

Forexamplead_type_id{1:'Emergency',2:'Urgent',3:'Elective',4:'Newborn',5:'Not Available',6:'NULL',7:'Trauma Center',8:'Not Mapped'}

```
df.admission_type_id = df.admission_type_id.apply(lambda x:ad_type_id[x])
df.discharge_disposition_id = df.discharge_disposition_id.apply(lambda x:discharge_disposition_id[x])
df.admission_source_id = df.admission_source_id.apply(lambda x:admission_sid[x])
```

```
df.head()
```

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	...
0	2278392	8222157	Caucasian	Female	[0-10)	?	NULL	Not Mapped	Physician Referral	1	...
1	149190	55629189	Caucasian	Female	[10-20)	?	Emergency	Discharged to home	Emergency Room	3	...
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	Emergency	Discharged to home	Emergency Room	2	...
3	500364	82442376	Caucasian	Male	[30-40)	?	Emergency	Discharged to home	Emergency Room	2	...
4	16680	42519267	Caucasian	Male	[40-50)	?	Emergency	Discharged to home	Emergency Room	1	...

The admission_type_id, discharge_disposition_id, admission_source_id, are all IDs we will not include them while creating the model. We will drop these columns after removing the Hospice and expired data.

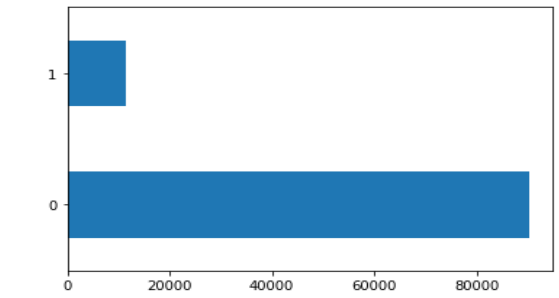
Target variable Imputation:

we are going to impute ">30" and "NO" with "0" and" <30" with "1". The unbalanced scale will increase exponentially in favor of "0". From above graph we can say that target variable is very imbalanced with 89% occurrences of "0"s and 11% of "1"s. Later we can use smote to balance the variable.

```
re_dict = {'>30':0,'NO':0,'<30':1}
df['readmitted_new'] = df.readmitted.replace(re_dict)
display(np.round(df.readmitted_new.value_counts()/df.shape[0]*100))
df.readmitted_new.value_counts().plot(kind='barh')
```

```
0    89.0
1    11.0
Name: readmitted_new, dtype: float64
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x20a73176be0>
```



Feature selection

Feature selection is a technique where we choose those features in our data that contribute most to the target variable. In other words, we choose the best predictors for the target variable.

All the variables are not significant enough. We can do feature selection using statistical method like ANNOVA F test, with the help of correlation, Business understanding or we can select using SelectKbest. Here we are using SelectKbest

```
from sklearn.feature_selection import SelectKBest, chi2
skb = SelectKBest(chi2).fit(x,y)
pd.DataFrame((x.columns,skb.scores_)).T.sort_values(by=1,ascending=False)
```

- Hospice mean a home providing care for the sick or terminally ill. As we are only concerned about the patient who were admitted in the hospital, we will remove data with Hospice. Also, there are few patients who have died, and for obvious reason they won't be admitted again. So, we will remove that data with it too.

- The admission_type_id, discharge_disposition_id, admission_source_id, are all IDs we will not include them while creating the model. We will drop these columns after removing the Hospice and expired data.

Train and test data preparation:

We have created two separate data frames for dependent and independent variables and split the dataset into 4 datasets (2 for independent variables and 2 for dependent variable) randomly

```
y = df.readmitted_new
x = df.drop('readmitted_new',axis=1)
print(x.shape)
x = pd.get_dummies(x)
print(x.shape)
```

```
(97825, 42)
(97825, 195)
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=.7,random_state=1)
```

Model building:

We will start with logistic regression analysis and subsequently check the model performance. We will train and validate our results for following models

1. Logistic Regression.
2. Decision Tree.
3. Random Forest.
4. SVM

	Model	Train set Score	Accuracy score	Precision	recall	f1_score	Cohen Kappa Score	MCC
0	Logistic Regression	0.886721	0.881764	0.472000	0.017037	0.032887	0.024870	0.071767
1	DecisonTree	1.000000	0.796647	0.169089	0.184811	0.176600	0.060861	0.060939
2	Random Forest	0.979803	0.881287	0.419847	0.015882	0.030607	0.022196	0.062652

Model performance with test data:

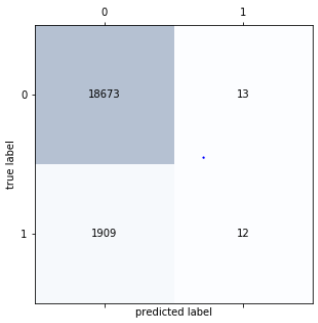
We will fit the model on test data. After training and testing our model we get the observation that

Dataset is highly imbalanced and will be biased towards the label which is more frequent and hence the evaluation matrices will not be accurate. There are lots of features created after one hot encoding which may be correlated and won't contribute to the prediction.

```
Classification report of LogisticRegression

              precision    recall  f1-score   support

           0       0.91      1.00      0.95     18686
           1       0.48      0.01      0.01      1921

    accuracy                           0.91     20607
   macro avg       0.69      0.50      0.48     20607
weighted avg       0.87      0.91      0.86     20607

============================================================
Classification report of DecisionTree

              precision    recall  f1-score   support

           0       0.91      0.90      0.90     18686
           1       0.13      0.15      0.14      1921

    accuracy                           0.83     20607
   macro avg       0.52      0.52      0.52     20607
weighted avg       0.84      0.83      0.83     20607

============================================================
Classification report of RandomForestClassifier

              precision    recall  f1-score   support

           0       0.91      1.00      0.95     18686
           1       0.26      0.01      0.01      1921

    accuracy                           0.91     20607
   macro avg       0.58      0.50      0.48     20607
weighted avg       0.85      0.91      0.86     20607
```

	Model	Train set Score	Accuracy score	Precision	recall	f1_score	Cohen Kappa Score	MCC
0	Logistic Regression	0.910549	0.906731	0.480000	0.006247	0.012333	0.009962	0.046364
1	DecisonTree	0.999979	0.827146	0.123451	0.140031	0.131220	0.035667	0.035754
2	Random Forest	0.981552	0.905712	0.288462	0.007808	0.015205	0.010342	0.033776

```
Train Score       0.9105486460629757
Test Score        0.9067307225700005
Precision Score   0.48
Recall Score      0.006246746486205101
F1_score          0.012332990750256937
Cohen Kapp Score  0.009961661957109014
Matthew Score     0.04636414981532377
ROC AUC Score     0.502775519234754
============================================================
confusion matric
```



Further Scope of Improvement:

- We could work on diag_1, diag_2, diag_3 feature. We can restrict them from creating further dimensions using one hot encoding by re-categorizing it.
- By using oversampling or SMOTE we can balance the target variable since our target variable is highly imbalanced.
- We will use hyperparameter tuning for selecting the best parameter while creating models using Ensemble techniques and find out important features from it.
- We will be using feature selection technique such as k-best to select the top 20 features for building the model and evaluating using appropriate performance matrices.