DIABETES READMISSION WITHIN 30 DAYS PROJECT REPORT

INTRODUCTION:

The goal of this project is to develop a machine learning model that can predict the likelihood of a patient being readmitted to the hospital within 30 days of discharge. This is a real-world problem that can have significant impacts on patient outcomes and healthcare costs. The dataset provided for this project contains information on more than 100,000 patient hospitalizations

Steps involved in Diabetes Readmission Project:

The workflow to build the end-to-end Machine Learning Project to predict Diabetes is as follows-

1. Importing required libraries &Collection of data
2. Cleaning the data
3. Data Visulization
4. Checking the Outliers
5. Performing Label Encoding
6. Checking Skewness
7. Checking Corelation
8. Model building and Evaluation
9. Plot Feature Importance
10. Conclusion

1. Importing required libraries and collection of data :

The very first step is to import the libraries and then collect the data

Importing all the required libraries

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Now, we will collect data using the *read_csv* method in the pandas library

In [2]:
```
df=pd.read_csv(r'Z:\diabetic_data.csv')
```

In [3]:
```
df
```

Out[3]:

	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)	?	6	25	1	1
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1
...	...	...	...	...	...	...	...	...	...	...
101761	443847548	100162476	AfricanAmerican	Male	[70-80)	?	1	3	7	3

There are 101766 rows in the dataset and 50 attributes.

Loading mapping file

In [6]:
```
maps=pd.read_csv(r'Z:\IDs_mapping.csv')
```

In [7]:
```
maps
```

Out[7]:

	admission_type_id	description
0	1	Emergency
1	2	Urgent
2	3	Elective
3	4	Newborn
4	5	Not Available
...	...	...
62	22	Transfer from hospital inpt/same fac reslt in...
63	23	Born inside this hospital
64	24	Born outside this hospital
65	25	Transfer from Ambulatory Surgery Center
66	26	Transfer from Hospice

67 rows × 2 columns

FEATURES

Encounter ID--- Unique identifier of an encounter

Patient number--- Unique identifier of a patient

Race Values: Caucasian, Asian, African American, Hispanic, and other

Gender Values: male, female, and unknown/invalid

Age Grouped in 10-year intervals: 0, 10), 10, 20), …, 90, 100)

Weight Weight in pounds

Admission type Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available

Discharge disposition Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available

Admission source Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital

Time in hospital Integer number of days between admission and discharge

Payer code Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay Medical

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

Number of lab procedures Number of lab tests performed during the encounter

Number of procedures Numeric Number of procedures (other than lab tests) performed during the encounter

Number of medications Number of distinct generic names administered during the encounter

Number of outpatient visits Number of outpatient visits of the patient in the year preceding the encounter

Number of emergency visits Number of emergency visits of the patient in the year preceding the encounter

Number of inpatient visits Number of inpatient visits of the patient in the year preceding the encounter

Diagnosis 1 The primary diagnosis (coded as first three digits of ICD9); 848 distinct values

Diagnosis 2 Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values

Diagnosis 3 Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values

Number of diagnoses Number of diagnoses entered to the system 0%

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

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.

Change of medications Indicates if there was a change in diabetic medications (either dosage or generic name). + + Values: “change” and “no change”

Diabetes medications Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”

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

Target Variable:

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 readmissionFeatures

check for total records, nulls, and data types of each attribute:

```
In [4]:  1 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                      101766 non-null  object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    101766 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                101766 non-null  object
 11  medical_specialty         101766 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                    101766 non-null  object
 19  diag_2                    101766 non-null  object
 20  diag_3                    101766 non-null  object
 21  number_diagnoses          101766 non-null  int64
 22  max_glu_serum             101766 non-null  object
```

```
 20  diag_3                    101766 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)
```

2. DATA CLEANING:

2.1: Checking if there are any duplicate values

```
In [10]: df[df.duplicated()]
```

Out[10]:

encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	...	citoglipton	insu

0 rows × 50 columns

2.2: Change '?' race to 'Unknown'

```
In [12]: df.replace({'race':{'?':'Unknown'}},inplace=True)
```

2.3: drop rows with unknown gender

```
In [13]: df = df[df.gender != 'Unknown/Invalid']
```

2.4: replacing the range of age group with its median value

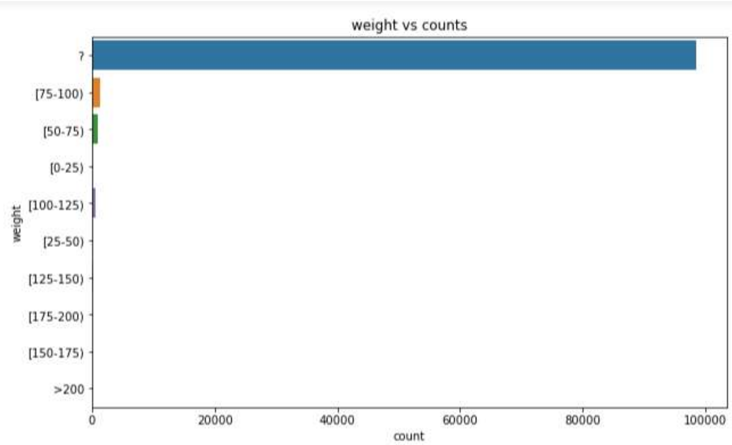
```
In [14]: df['age'].unique()
```

```
Out[14]: array(['[0-10)', '[10-20)', '[20-30)', '[30-40)', '[40-50)', '[50-60)',
               '[60-70)', '[70-80)', '[80-90)', '[90-100)'], dtype=object)
```

```
In [15]:
df=df.replace({'age':{'[0-10)':'5'}})
df=df.replace({'age':{'[10-20)':'15'}})
df=df.replace({'age':{'[20-30)':'25'}})
df=df.replace({'age':{'[30-40)':'35'}})
df=df.replace({'age':{'[40-50)':'45'}})
df=df.replace({'age':{'[50-60)':'55'}})
df=df.replace({'age':{'[60-70)':'65'}})
df=df.replace({'age':{'[70-80)':'75'}})
df=df.replace({'age':{'[80-90)':'85'}})
df=df.replace({'age':{'[90-100)':'95'}})
```

2.5: Drop attributes with high percentage of Nulls

weight column has 96% null values, payer_code has 52% null values, medical_speciality has 53% null values



Drop attributes with high percentage of Nulls

weight column has 96% null values, payer_code has 52% null values, medical_speciality has 53% null values

```
In [17]: df.drop(['weight', 'payer_code', 'medical_specialty'],axis =1, inplace=True)
```

2.6: delete attributes with same values

```
In [18]: df.drop(['citoglipton','examide'], axis =1, inplace=True)
```

2.7: Removing all the data where the patient is Expired(dead) (at home or in hospital)

In [23]:

```
df = df[df.discharge_disposition_id != 11]
df = df[df.discharge_disposition_id != 19]
df = df[df.discharge_disposition_id != 20]
df = df[df.discharge_disposition_id != 21]
```

2.8: Change admission type urgent and trauma into emergency

[24]:

```
df['admission_type_id'].replace([2,7],1,inplace=True)
```

2.9: Change admission type Null and not mapped into not available

In [25]:

```
df['admission_type_id'].replace([6,8],5,inplace=True)
```

2.10: putting in all the discharge type transfer to home in id 1 (which also means move/transfer to home)

In [26]:

```
df['discharge_disposition_id'].replace([6,8,13],1,inplace=True)
```

2.11: putting in all types of tranfer together

]:

```
df['discharge_disposition_id'].replace([3,4,5,10,15,16,17,22,23,24,27,28,29,30],2,inplace=True)
```

2.12: Replacing discharge type not mapped,unknown to NaN

[28]:

```
df['discharge_disposition_id'].replace([25,26],18,inplace=True)
```

2.13: putting in all the referral data together with the help of replace function (for admission source)

```
In [29]:  df['admission_source_id'].replace([2,3],1,inplace=True)
```

2.14: putting in all the transfer data together (for admission source)

```
In [30]:  df['admission_source_id'].replace([5,6,10,18,25,26],4,inplace=True)
```

2.15: putting in all the not available data together (for admission source)

```
In [31]:  df['admission_source_id'].replace([15,17,20,21],9,inplace=True)
```

2.16: no and >30 convert to 0 since both mean that person will not be readmitted 'within' 30 days and for <30 it means person was readmitted within 30 days

```
]:  df.replace({'readmitted': {'NO': 0, '>30': 0, '<30': 1 }}, inplace=True)
```

2.17: eliminating the data where diag1,diag2,diag3 all this 3 values are missing

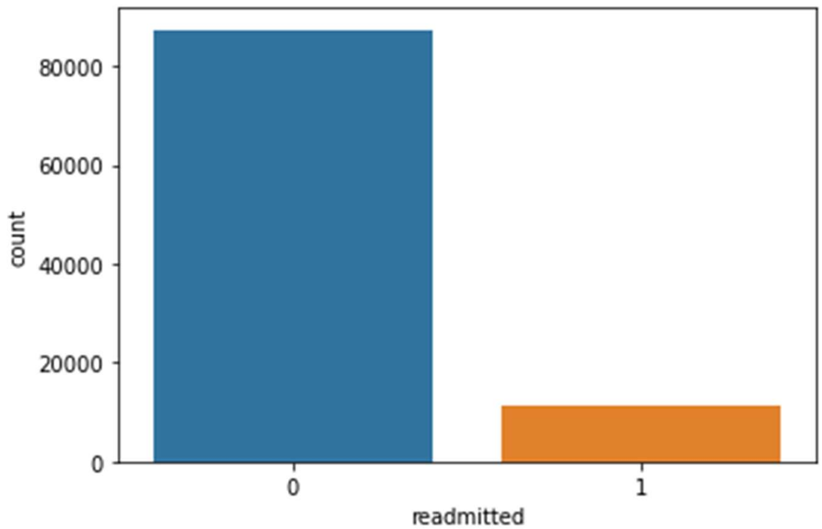
```
In [36]:  diag_missing=df[(df['diag_1'] == '?' ) | (df['diag_2'] == '?' ) | (df['diag_3'] == '?')].index

In [37]:  diag_missing
Out[37]: Int64Index([     0,     66,    192,    194,    216,    249,    263,    283,
               286,    294,
            ...
             99702,  99979, 100559, 100787, 101173, 101192, 101278, 101474,
            101560, 101719],
           dtype='int64', length=1520)

In [38]:  df.drop(diag_missing,inplace=True)

In [39]:  df.shape
Out[39]: (98591, 45)
```

There are no null value , there are no duplicates and our data is now entirely clean, now further we will perform visulization and get some insights from our data

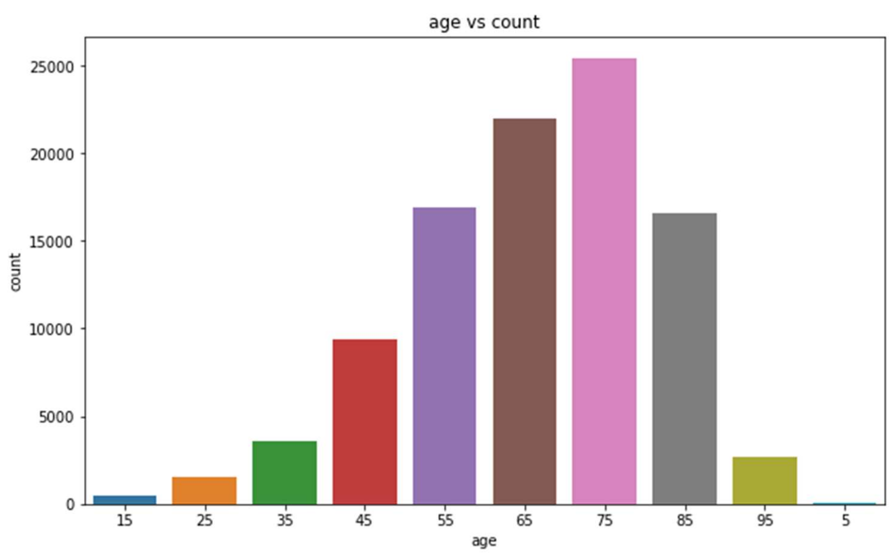
```
In [41]:  1  df.isna().sum()

Out[41]: encounter_id                0
         patient_nbr                 0
         race                        0
         gender                      0
         age                         0
         admission_type_id           0
         discharge_disposition_id    0
         admission_source_id         0
         time_in_hospital            0
         num_lab_procedures          0
         num_procedures              0
         num_medications             0
         number_outpatient           0
         number_emergency            0
         number_inpatient            0
         diag_1                      0
         diag_2                      0
         diag_3                      0
         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
```

```
diag_3                      0
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
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
```

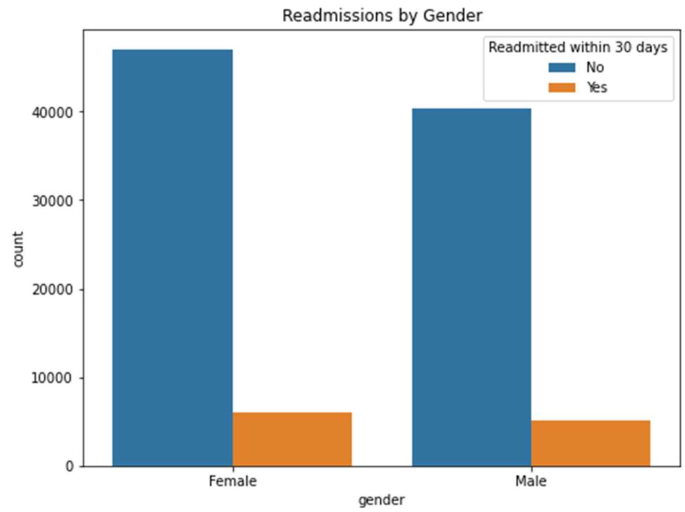
3. DATA VISULIZATION:



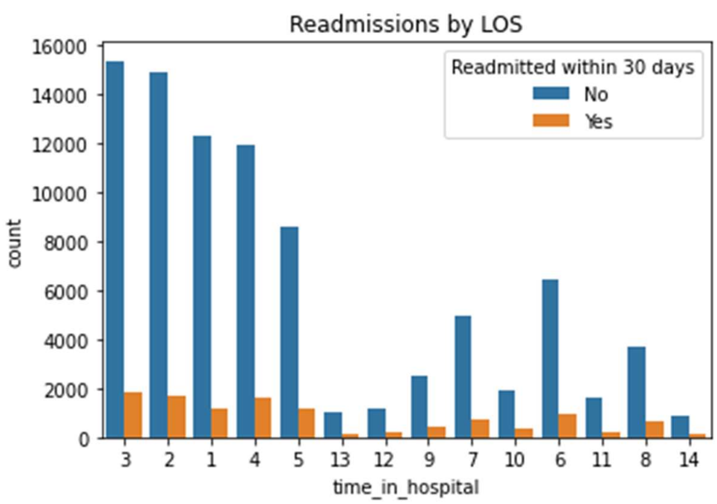
Insight from above graph: Our target variable is imbalance. Number of readmitted patient are quite less as compared to Not readmitted



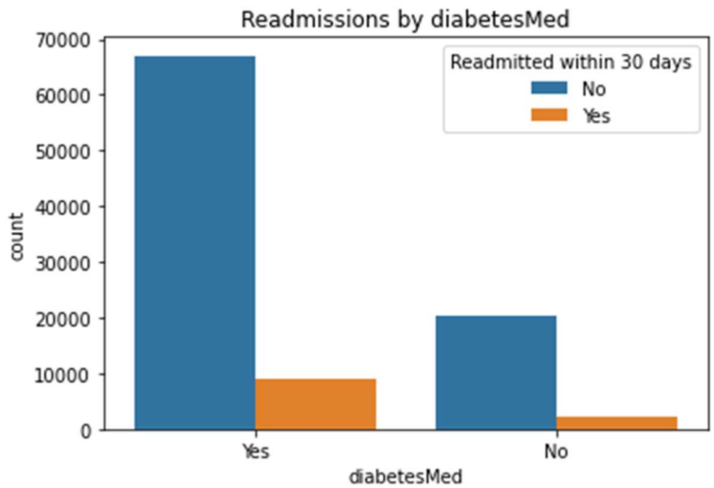
Insight from above graph:people with age group 75 have maximum count



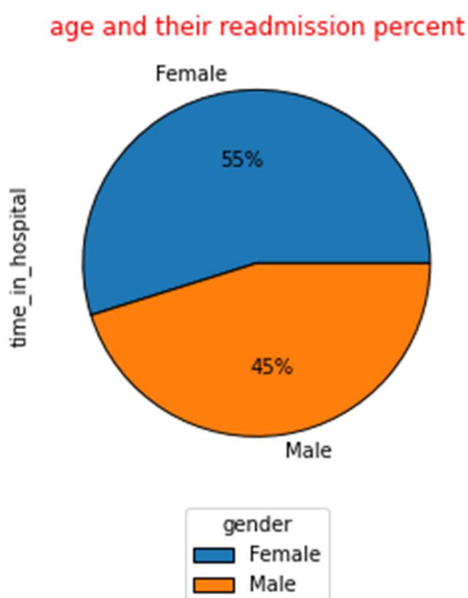
Insight from above graph: from the above graph we can say that the readmission rate of female is more as compared to male



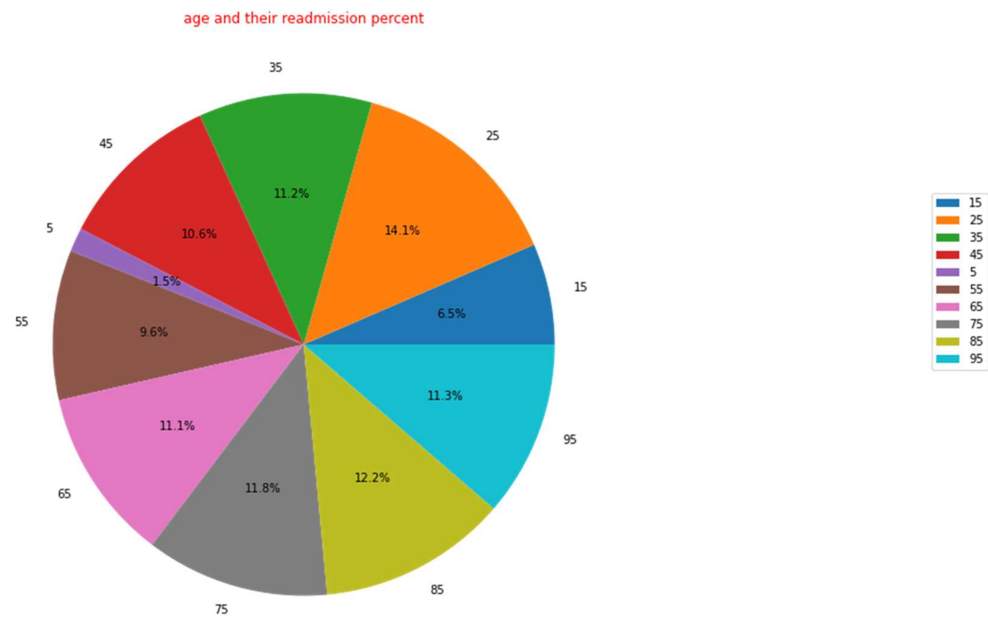
Insight from above graph: Mostly patient between 2 to 4 days are admitted frequently



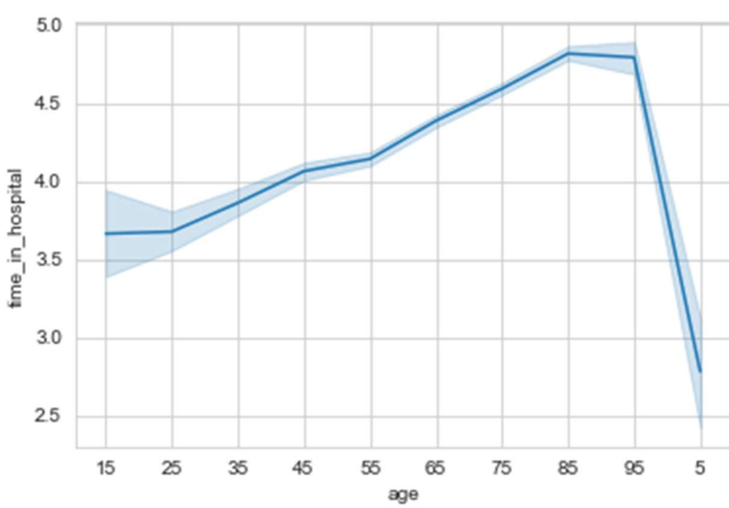
Insight from above graphs: Patients provided with diabetes medication are readmitted often



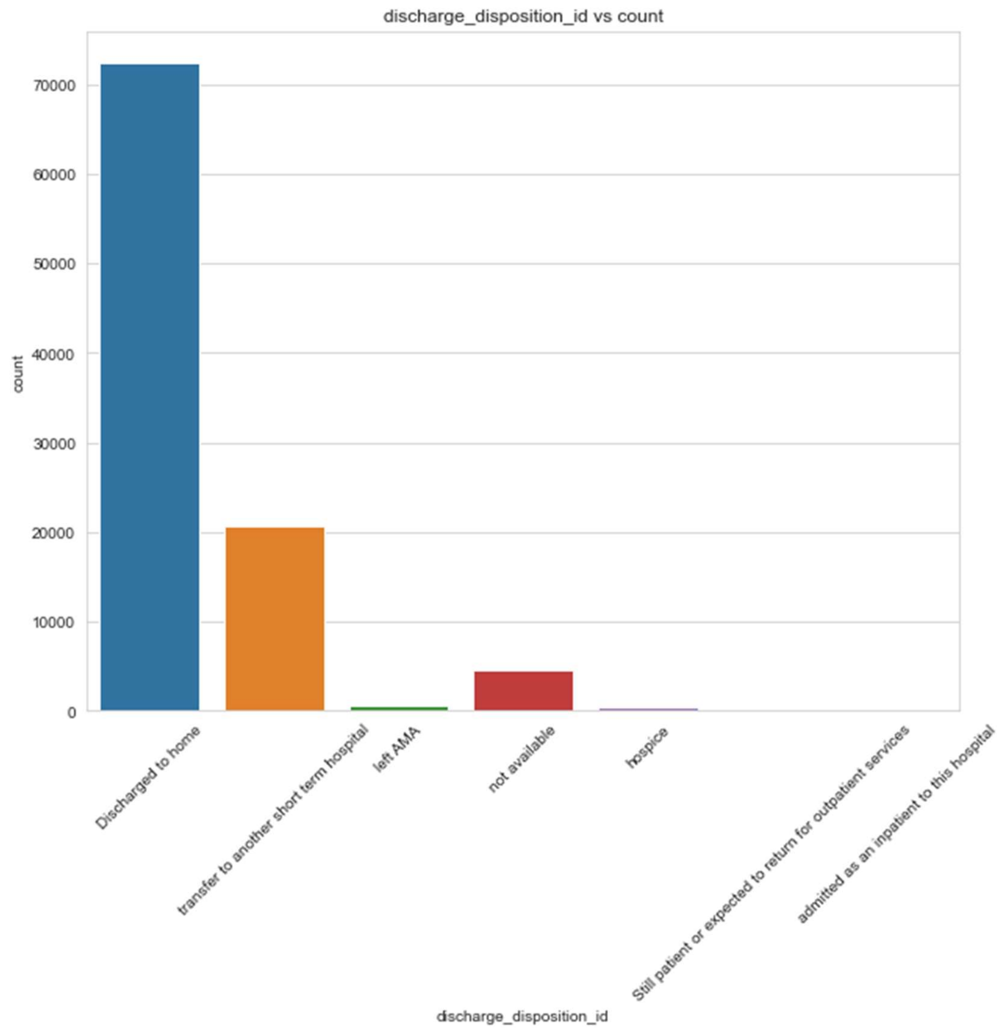
Insight from above graph: Female have spend more time in hospital



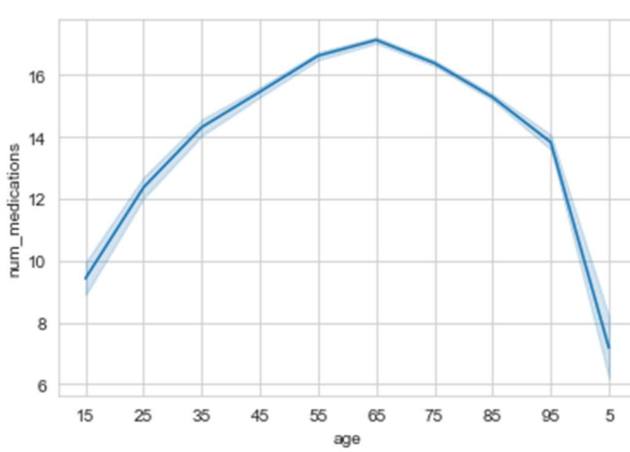
Insight from above graph: From age feature we can see that majority of patients have higher age(ie 85,75,65,55) , age can be one of the factors for early readmission



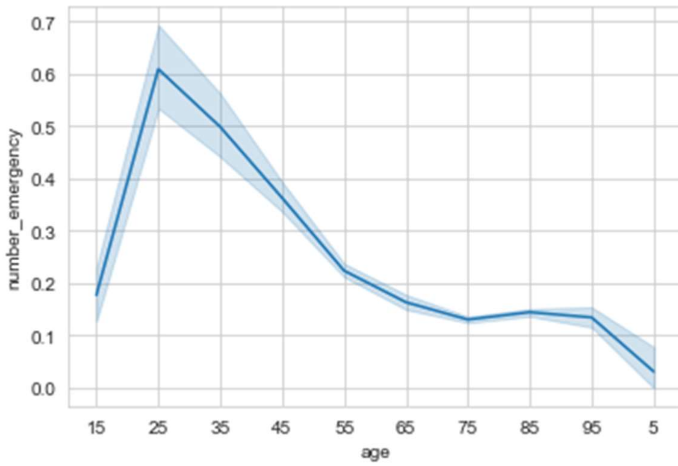
Insight from above graph: The age group between 85 to 95 have spend maximum time in hospital



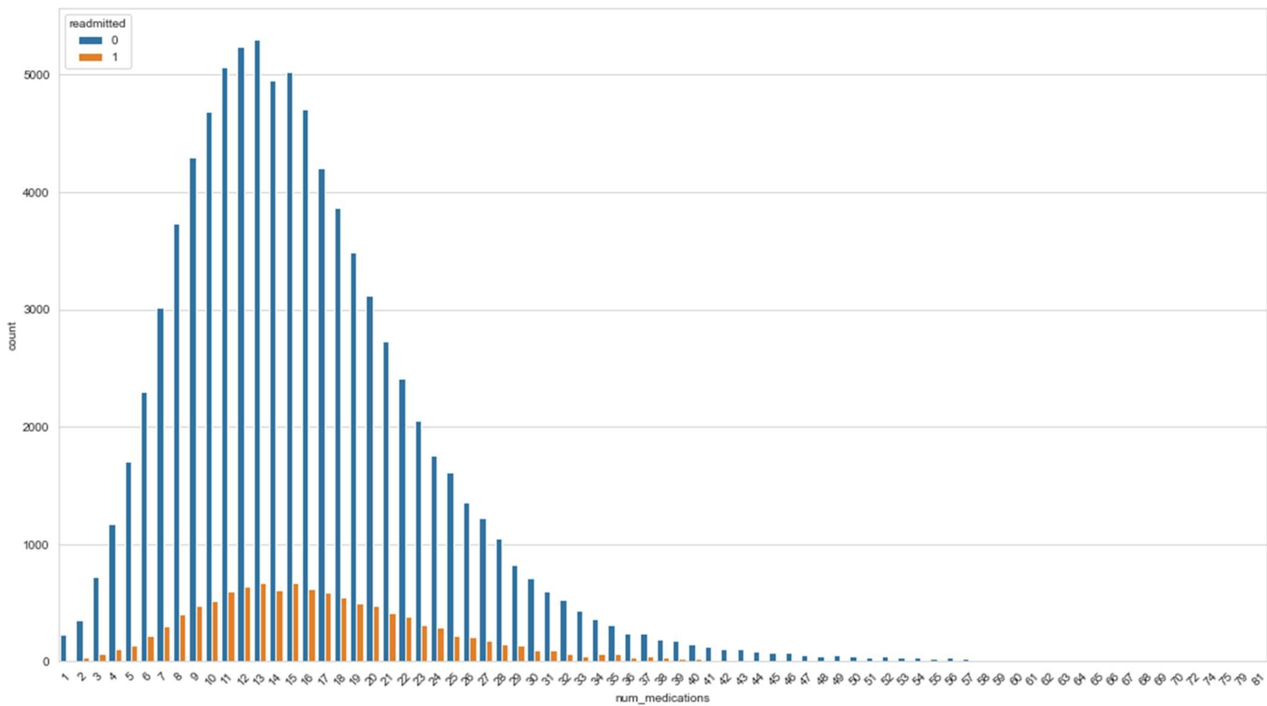
Insight from above graph: Discharge id 1 ie dischage to home is the highest in our data followed by id 2 ie transfer to another short term hospital



Insight from above graph:maximum number of medicatons are take by age group 65



Insight from above graph:maximum number of emergecy were at the age of 25



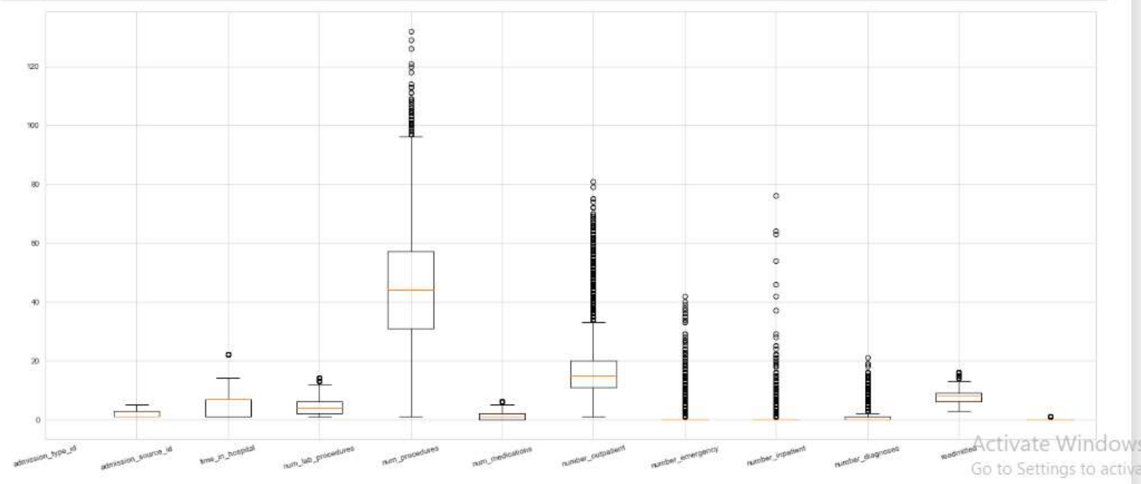
Insight from above graph:as we can see number of medications between 9 to 20 have high chance of readmission

4. CHECKING THE OUTLIERS

Droping the columns that wont play any role in telling us weather the patient will be readmitted or not

```
[79]: new_df.drop(['encounter_id','patient_nbr','race'],axis=1,inplace=True)
```

```
In [88]: plt.figure(figsize=(24, 10))
         plt.boxplot(df_numeric)
         plt.xticks(range(len(df_numeric.columns.values)),df_numeric.columns.values, rotation = 15)
         plt.show()
```



```python
for k, v in df_numeric.items():
        q1 = v.quantile(0.25)
        q3 = v.quantile(0.75)
        IQR = q3 - q1
        v_col = v[(v < q1 - 1.5 * IQR) | (v > q3 + 1.5 * IQR)]
        perc = np.shape(v_col)[0] * 100.0 / np.shape(new_df)[0]
        print(" %s outliers = %.2f%%" % (k, perc))
```

```
admission_type_id outliers = 0.00%
admission_source_id outliers = 0.01%
time_in_hospital outliers = 2.21%
num_lab_procedures outliers = 0.12%
num_procedures outliers = 4.87%
num_medications outliers = 3.32%
number_outpatient outliers = 16.60%
number_emergency outliers = 11.25%
number_inpatient outliers = 6.93%
number_diagnoses outliers = 0.06%
readmitted outliers = 11.41%
```

As from the above percentage code for outliers we can see that there are no such extreme outilers

5. Performing Label Encoding

In [99]:
```python
cat_data=df2.select_dtypes('object')
```

n [100]:
```python
cat_data.columns
```

ut[100]:
```
Index(['gender', 'age', 'discharge_disposition_id', 'diag_1', 'diag_2',
       'diag_3', 'max_glu_serum', 'A1Cresult', 'metformin', 'repaglinide',
       'nateglinide', 'chlorpropamide', 'glimepiride', 'acetohexamide',
       'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone',
       'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone', 'tolazamide',
       'insulin', 'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-rosiglitazone',
       'metformin-pioglitazone', 'change', 'diabetesMed'],
      dtype='object')
```

n [101]:
```python
from sklearn.preprocessing import LabelEncoder

```

n [102]:
```python
lbe=LabelEncoder()
```

n [103]:
```python
for colu in df2:
    df2[colu]=lbe.fit_transform(df2[colu])
```

In [104]:
```python
df2
```

Out[104]:

	gender	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	num_medicatic
1	0	0	0	0	2	2	58	0	
2	0	1	0	0	2	1	10	5	
3	1	2	0	0	2	1	43	1	
4	1	3	0	0	2	0	50	0	
5	1	5	0	0	0	2	30	6	
...	...	...	...	...	...	...	...	...	
101761	1	7	0	6	2	2	50	0	
101762	0	8	0	6	1	4	32	3	
101763	1	7	0	0	2	0	52	0	
101764	0	8	0	6	2	9	44	2	
101765	1	7	0	0	2	5	12	3	

98591 rows × 42 columns

we have perform label encoding on dataframe df2

6. CHECKING SKEWNESS:

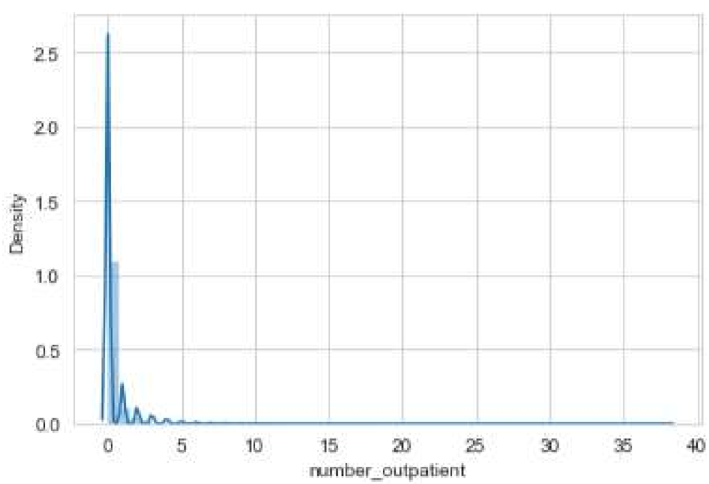
There are 3 techniques to remove skewness:

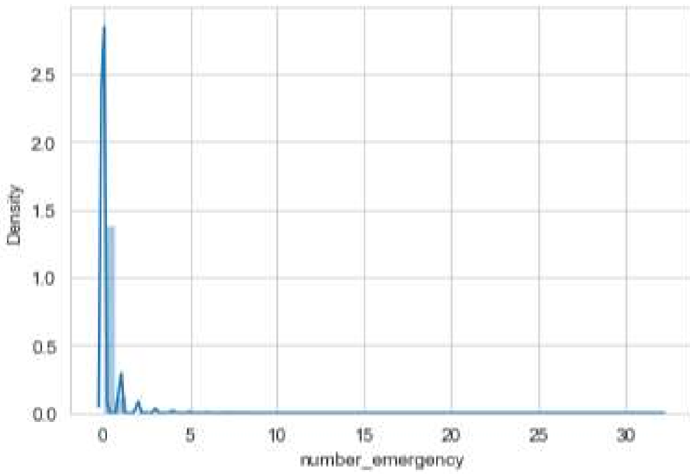
1)log method

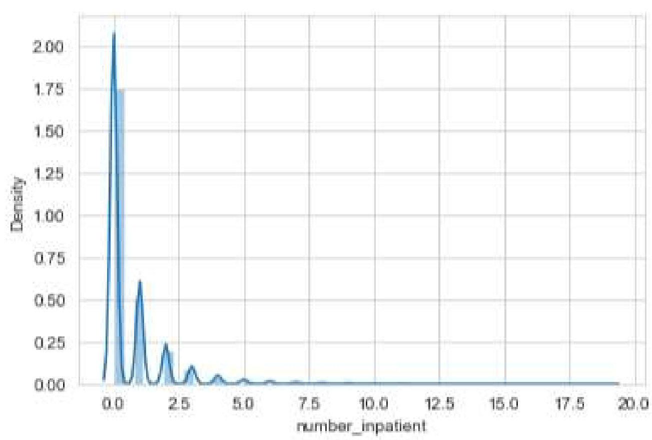
2)sqrt method

3)boxcox

```
In [105]:  df_numeric_data=df2.select_dtypes('int64')

In [106]:  df_numeric_data.columns

Out[106]: Index(['admission_type_id', 'admission_source_id', 'time_in_hospital',
                 'num_lab_procedures', 'num_procedures', 'num_medications',
                 'number_outpatient', 'number_emergency', 'number_inpatient',
                 'number_diagnoses', 'readmitted'],
                dtype='object')

In [107]:  for cole in df_numeric_data:
               print(f'Skewness Value is {skew(df_numeric_data[cole])}')
               sns.distplot(df_numeric_data[cole])
               plt.show()
```







As we can see from the graphs above discharge_disposition_id,number_outpatient,number_inpatient and number_emergency are highly skewed

```
In [108]:  df2['discharge_disposition_id'].skew()
Out[108]: 1.095852316269492
```

```
In [109]:  df2['number_outpatient'].skew()
Out[109]: 8.328007394938435
```

```
In [110]:  df2['number_inpatient'].skew()
Out[110]: 3.563823046067838
```

```
In [111]:  df2['number_emergency'].skew()
Out[111]: 11.499243530110888
```

we will try to reduce the skewness using sqrt method

```
In [112]:  df2['discharge_disposition_id']=np.sqrt(df2['discharge_disposition_id'])

In [113]:  df2['number_inpatient']=np.sqrt(df2['number_inpatient'])

In [114]:  df2['number_outpatient']=np.sqrt(df2['number_outpatient'])

In [115]:  df2['number_emergency']=np.sqrt(df2['number_emergency'])

In [116]:  df2['discharge_disposition_id'].skew()
Out[116]: 1.0700682703184108

In [117]:  df2['number_inpatient'].skew()
Out[117]: 1.2942397306360132

In [118]:  df2['number_outpatient'].skew()
Out[118]: 2.707205121696364

In [119]:  df2['number_emergency'].skew()
Out[119]: 3.481361864305842
```

7. CHECKING CORRELATION

If the degree of correlation between variables is high enough, it can cause problems when you fit the model and interpret the results. Generally, a correlation with an absolute value around 0.7-0.8 or higher is considered a high correlation

In [121]:
```
df_numeric_data.corr().style.background_gradient()
```

Out[121]:

	admission_type_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_outpatient
admission_type_id	1.000000	0.112208	-0.022459	-0.127644	0.082489	0.075657	0.048091
admission_source_id	0.112208	1.000000	-0.004372	0.094942	-0.166643	-0.073173	0.021342
time_in_hospital	-0.022459	-0.004372	1.000000	0.320708	0.187819	0.461766	-0.011041
num_lab_procedures	-0.127644	0.094942	0.320708	1.000000	0.051856	0.265977	-0.007818
num_procedures	0.082489	-0.166643	0.187819	0.051856	1.000000	0.378938	-0.027008
num_medications	0.075657	-0.073173	0.461766	0.265977	0.378938	1.000000	0.044056
number_outpatient	0.048091	0.021342	-0.011041	-0.007818	-0.027008	0.044056	1.000000
number_emergency	-0.017631	0.074900	-0.010097	-0.000196	-0.043071	0.015009	0.100997
number_inpatient	-0.032426	0.049729	0.072782	0.038620	-0.069119	0.062729	0.108501
number_diagnoses	-0.099211	0.099071	0.211114	0.148498	0.055761	0.243466	0.094071
readmitted	-0.010580	0.011090	0.044427	0.022708	-0.011376	0.038842	0.018789

as we can see from the above graph there is no high correlation

8. MODEL BUILDING AND EVALUATION

Goal is to predict with the help of given features whether the patient will be readmitted within 30 days or not

splitting x and y

```
In [122]: x=df2.iloc[0:,0:-1]
```

```
In [123]: x
Out[123]:
```

	gender	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	num_medicatic
1	0	0	0	0.00000	2	2	58	0	
2	0	1	0	0.00000	2	1	10	5	
3	1	2	0	0.00000	2	1	43	1	
4	1	3	0	0.00000	2	0	50	0	
5	1	5	0	0.00000	0	2	30	6	
...	...	...	...	...	...	...	...	...	
101761	1	7	0	2.44949	2	2	50	0	
101762	0	8	0	2.44949	1	4	32	3	
101763	1	7	0	0.00000	2	0	52	0	
101764	0	8	0	2.44949	2	9	44	2	
101765	1	7	0	0.00000	2	5	12	3	

98591 rows × 41 columns


```
In [124]: y=df2.iloc[0:,-1]
```

```
In [125]: y
Out[125]: 1         0
          2         0
          3         0
          4         0
          5         0
                   ..
          101761    0
          101762    0
          101763    0
          101764    0
          101765    0
          Name: readmitted, Length: 98591, dtype: int64
```

splitting x into xtrain and xtest splitting y into ytrain and ytest and then perform train_test_split ,test_size is how much amount of data we want to put in for testing ,we perform stratify to balance out our data

```
In [126]:  from sklearn.model_selection import train_test_split
           xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1,stratify=y)
```

Classification report-A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model

```
In [127]:  from sklearn.metrics import classification_report
```

we will perform LogisticRegression on our model

logistic regression-Logistic regression is an example of supervised learning,It is used to calculate or predict the probability of a binary (yes/no) event occurring.

```
In [128]:  from sklearn.linear_model import LogisticRegression

In [129]:  logreg=LogisticRegression()

In [130]:  logreg.fit(xtrain,ytrain)
Out[130]:  LogisticRegression()

In [131]:  ypred=logreg.predict(xtest)

In [132]:  ypred
Out[132]:  array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

accuracy_score- is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made

confusion_matrix- A confusion matrix is a table that is used to define the performance of a classification algorithm.

roc_auc_score- roc_auc_score is defined as the area under the ROC curve, which is the curve having False Positive Rate on the x-axis and True Positive Rate on the y-axis at all classification thresholds

In [133]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score
ac=accuracy_score(ytest,ypred)
print(f'accuracy score is {ac}')
```

```
accuracy score is 0.8857596862532964
```

In [134]:
```
train_score=logreg.score(xtrain,ytrain)
test_score=logreg.score(xtest,ytest)
print(f'Training score is {train_score}')
print(f'Testing score is {test_score}')
print(classification_report(ytest,ypred))
```

```
Training score is 0.8858910640024343
Testing score is 0.8857596862532964
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     26203
           1       0.00      0.00      0.00      3375

    accuracy                           0.89     29578
   macro avg       0.44      0.50      0.47     29578
weighted avg       0.78      0.89      0.83     29578
```

we have got great train and test score (low bias and low variance) and there is no overfitting and no underfitting
now lets check the confusion_matrix
Table that describes the performance of a classification model.

True Positives (TP): we correctly predicted that they do have diabetes

True Negatives (TN): we correctly predicted that they don't have diabetes

False Positives (FP): we incorrectly predicted that they do have diabetes (a "Type I error")

False Negatives (FN): we incorrectly predicted that they don't have diabetes (a "Type II error")

In [135]:
```
cm=confusion_matrix(ytest,ypred)
print(cm)
```

```
[[26199     4]
 [ 3375     0]]
```

as we can see type 2 error is very high ,so will try to reduce it

```
In [136]: ra=roc_auc_score(ytest,ypred)
          print(f'ROC_AUC_SCORE {ra}')

ROC_AUC_SCORE 0.49992367286188605

In [137]: print(f'actual values: {ytest[:80].values}')
          print(f'pred values: {ypred[:80]}')

actual values: [1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0]
pred values: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0]

In [138]: logreg.predict_proba(xtest)

Out[138]: array([[0.91882883, 0.08117117],
       [0.93285313, 0.06714687],
       [0.90346425, 0.09653575],
       ...,
       [0.90650734, 0.09349266],
       [0.87144362, 0.12855638],
       [0.9131244 , 0.0868756 ]])

In [139]: ypredprob=logreg.predict_proba(xtest)[:,1]
```

```
In [139]: ypredprob=logreg.predict_proba(xtest)[:,1]

In [140]: ypredprob

Out[140]: array([0.08117117, 0.06714687, 0.09653575, ..., 0.09349266, 0.12855638,
       0.0868756 ])

In [141]: from sklearn.preprocessing import binarize

In [142]: y_pred=binarize([ypredprob],threshold=0.1)[0]

In [143]: y_pred

Out[143]: array([0., 0., 0., ..., 0., 1., 0.])

In [144]: ac=accuracy_score(ytest,y_pred)
          cm=confusion_matrix(ytest,y_pred)
          print(ac)
          print(cm)

0.529346135641355
[[13366 12837]
 [ 1084  2291]]
```

now as we can see after performing the required steps , type 2 error has reduced

we have got great train and test score (low bias and low variance) and there is no overfitting and no underfitting ,but as we can see precision,recall,f1-score is 0.00 for 1 so we will have to use SMOTE teachnique to make it better

```
In [148]: pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\admin\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\admin\anaconda3\lib\site-packages (from imblearn) (0.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-learn->imble
arn) (2.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-learn->imblea
rn) (1.0.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.
7.3)
Requirement already satisfied: numpy>=1.17.3 in c:\users\admin\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.21.5)
Note: you may need to restart the kernel to use updated packages.
```

```
In [149]: from imblearn.over_sampling import SMOTE
```

```
In [150]: sm=SMOTE(random_state=2)
```

```
In [151]: X=df2.iloc[0:,0:-1]
```

```
In [152]: Y=df2.iloc[0:,-1]
```

```
In [153]: from sklearn.model_selection import train_test_split
```

```
In [154]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state = 2,stratify=Y)
```

```
In [155]: print( X_train.shape)
          print( Y_train.shape)
          print( X_test.shape)
          print( Y_test.shape)
```

```
(69013, 41)
(69013,)
(29578, 41)
(29578,)
```

```
In [156]: from sklearn.linear_model import LogisticRegression
```

```
In [157]: lg=LogisticRegression()
```

```
In [158]: lg.fit(X_train,Y_train)        #.ravel
```

```
Out[158]: LogisticRegression()
```

```
In [159]:
y_pred=lg.predict(X_test)
```

```
In [160]:
y_pred
```

```
Out[160]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [161]: print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     26203
           1       0.00      0.00      0.00      3375

    accuracy                           0.89     29578
   macro avg       0.44      0.50      0.47     29578
weighted avg       0.78      0.89      0.83     29578
```

```
In [162]: sm=SMOTE(random_state=2)
```

```
In [163]: X_train_res,Y_train_res=sm.fit_resample(X_train,Y_train)
```

```
In [164]: lg1=LogisticRegression()
```

```
In [164]: lg1=LogisticRegression()
```

```
In [165]: lg1.fit(X_train_res,Y_train_res)
```

```
Out[165]: LogisticRegression()
```

```
In [166]: pre=lg1.predict(X_test)
```

```
In [167]:
train_scores=lg1.score(X_train_res,Y_train_res)
test_scores=lg1.score(X_test,Y_test)
print(f'Training score is {train_scores}')
print(f'Testing score is {test_scores}')

print(classification_report(Y_test,pre))
```

```
Training score is 0.6999247603781609
Testing score is 0.6918317668537426
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

As we can see after performing smote teachnique we get better precision,recall,f1-score for 1

Now we will try various models on our data and select the most appropriate one

KNeighborsClassifier- is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point

DecisionTree- A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks

In [168]:
```python
def mymodel(model):
    model.fit(X_train_res,Y_train_res)
    ypred = model.predict(X_test)

    train = model.score(X_train_res,Y_train_res)
    test = model.score(X_test,Y_test)
    print(f'Trainig Score is {train}')
    print(f'Test Score is {test}')

    print(classification_report(Y_test,pre))
    return model
```

In [169]:
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [170]:
```python
# knn = mymodel(KNeighborsClassifier())
```

In [171]:
```python
# trainac = []
# testac = []

# for i in range(1,31):
#     knn = KNeighborsClassifier(n_neighbors=i)
#     knn.fit(X_train_res,Y_train_res)

#     Train = knn.score(X_train_res,Y_train_res)
#     Test = knn.score(X_test,Y_test)
#     trainac.append(Train)
#     testac.append(Test)
```

In [172]:
```python
LG = mymodel(LogisticRegression())
```

```
Trainig Score is 0.6999247603781609
Test Score is 0.6918317668537426
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

In [173]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [174]:
```
dt = mymodel(DecisionTreeClassifier())
```

```
Trainig Score is 1.0
Test Score is 0.7812563391710055
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

In [175]:
```
parameters = {
    'criterion':['gini','entropy'],
    'max_depth': list(range(1,20)),
    'min_samples_leaf':list(range(1,20))
}
```

Both gini and entropy are measures of impurity of a node

GridSearchCV is a technique for finding the optimal parameter values from a given set of parameters in a grid

In [176]:
```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(DecisionTreeClassifier(),parameters,verbose=2)
grid.fit(X_train_res,Y_train_res)
```

```
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=4; total time=180.0min
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=4; total time=   1.7s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=4; total time=   1.2s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=4; total time=   1.7s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=4; total time=   2.6s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=5; total time=   1.5s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=5; total time=   1.6s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=5; total time=   1.4s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=5; total time=   1.3s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=5; total time=303.3min
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=6; total time=   2.8s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=6; total time=   2.7s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=6; total time=   4.4s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=6; total time=   2.6s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=6; total time=   2.5s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=7; total time=   2.4s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=7; total time=   2.4s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=7; total time=   2.9s
[CV] END criterion=entropy, max_depth=17, min_samples_leaf=7; total time=   2.3s
```

In [177]:
```
grid.best_score_
```

Out[177]: 0.8847787776136725

In [178]:
```
grid.best_estimator_
```
Out[178]:
```
DecisionTreeClassifier(max_depth=14, min_samples_leaf=2)
```
In [179]:
```
dt = mymodel(grid.best_estimator_)
```
```
Trainig Score is 0.9110618600543033
Test Score is 0.8444451957536007
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

SVM-A support vector machine (SVM) is a type of deep learning algorithm that performs supervised learning for classification or regression of data groups

SVC- Support Vector Classifier, is a supervised machine learning algorithm typically used for classification tasks

In [180]:
```
from sklearn.svm import SVC
```
In [181]:
```
svm = SVC()
```
In [182]:
```
svm.fit(X_train_res,Y_train_res)
```
Out[182]:
```
SVC()
```
In [183]:
```
trains = svm.score(X_train_res,Y_train_res)
tests = svm.score(X_test,Y_test)
print(f'Accuracy Score of train is {trains}')
print(f'Accuracy Score of test is {tests}')
```
```
Accuracy Score of train is 0.5809480192351729
Accuracy Score of test is 0.5643721685036176
```

Gaussian Naive Bayes (GNB)-is a classification technique used in Machine Learning (ML) based on the probabilistic approach and Gaussian distribution. Gaussian Naive Bayes assumes that each parameter (also called features or predictors) has an independent capacity of predicting the output variable.

Bernoulli Naive Bayes- is one of the variants of the Naive Bayes algorithm in machine learning. It is very useful to be used when the dataset is in a binary distribution where the output label is either present or absent

MultinomialNB-solves multiclass classification task where you have more than 2 categories in the target variable

```
In [184]:  1 from sklearn.naive_bayes import MultinomialNB,GaussianNB,BernoulliNB
```

```
In [185]:  1 bnb = mymodel(BernoulliNB())

Trainig Score is 0.7431384736170631
Test Score is 0.7115761714788018
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

```
In [186]:  1 mul = mymodel(MultinomialNB())

Trainig Score is 0.5594801923517289
Test Score is 0.5593346406112651
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

```
In [187]:  1 gaus = mymodel(GaussianNB())

Trainig Score is 0.6075190552520527
Test Score is 0.3274393130029076
              precision    recall  f1-score   support

           0       0.91      0.73      0.81     26203
           1       0.17      0.42      0.24      3375

    accuracy                           0.69     29578
   macro avg       0.54      0.57      0.52     29578
weighted avg       0.82      0.69      0.74     29578
```

As we can see logistic and BernoulliNB can be considered as best working models on our dataset

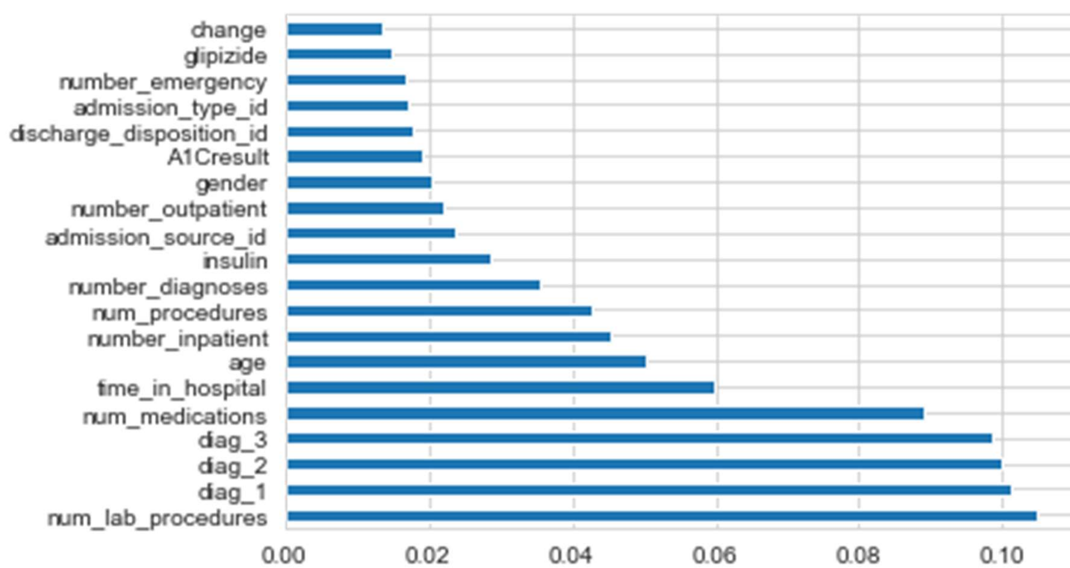
9. Plot Feature Importance

```
In [188]:  from sklearn.ensemble import RandomForestClassifier
           %matplotlib inline
```

```
In [189]:  mod=RandomForestClassifier()
           mod.fit(x,y)
```

```
Out[189]: RandomForestClassifier()
```

```
In [190]:  (pd.Series(mod.feature_importances_, index=x.columns)
              .nlargest(20)
              .plot(kind='barh'))
```



10.CONCLUSION

Six major features are found to have high impact on diabetes patient readmission: number of lab procedures,diag1,diag2,diag3, number of medications, time spent in hospital

The logistic regression classifier modeling achieves 0.69 accuracy and can be considered as the best for our dataset

To correctly predict the readmission , hospitals should carefully examine the clinical data of patients and pay special attention to the above major features.

Some other features might be worth collecting, for example,family history.

This analytic method can be applied to different diseases other than diabetes.

In conclusion, ML could help healthcare providers to identify those patients who are prone to short-term readmission and might reduce the probability of readmission within 30 days by altering the risk factors.

THE END