A

Capstone Project Report

On

"DIABETIC PATIENTS' READMISSION PREDICTION"

Submitted in partial fulfilment of the requirements for the award of degree

PGP in **Data Science and Engineering**

Affiliated to

Great Lakes Institute of Executive Learning

Guided by-

Mr. SRIKAR MUPPIDI

Submitted by-

- UMAIR AHMED E F
- TEJA LAKKADASU
- VISHAL KARTHICK R
- MANNE VASU RAJENDRA
- YOGESHWARAN V

ACKNOWLEGDGEMENT

We would like to thank our mentor Mr. Srikar Muppidi for providing his valuable guidance and suggestions over the course of our Project Work. We also thank him for his continuous encouragement and interest towards our Project work. We are extremely grateful to all our teaching and non-teaching staff members of GREAT LEARNING, who showed keen interest and inquired us about our development and progress.

We greatly admire and acknowledge the constant support we received from our friends and team members for all the effort and hard work that they have put into completing this project.

Table of Content

1. Abstract:

Adult patients with diabetes mellitus (DM) represent one-fifth of all unplanned hospital readmissions but some may be preventable through continuity of care with better DM self-management. Hospital readmissions increase the healthcare costs and negatively influence hospitals' reputation. Medicare counts as a readmission, if any of those patients who ended up back in any hospital for the same condition, except for planned returns like a second phase of surgery. A hospital will be penalized if its readmission rate is higher than expected given the national trends in any one of those categories and they will also lose insurance money since it is provided on an annual basis. Predicting readmissions in early stages allows prompting great attention to patients with high risk of readmission, which leverages the healthcare system and saves healthcare expenditures.

2. Introduction:

Diabetes is one of the major diseases across the US and even across the globe. 18% of the US GDP is spent on healthcare and a similar percentage of amount is spent in most of the developed countries. In the United States, type 2 diabetes is prevalent for certain groups other than for Caucasians. These people include:

- Native Americans
- African Americans
- Hispanics
- Asian Americans

The domain of this Capstone Project falls under the purview of “Healthcare Analytics”. The capstone project will present the analysis of a large clinical database (III) that was undertaken to examine the historical patterns of diabetes care in patients with diabetes admitted to a US hospital and to indicate future directions which will lead to a reduction in hospital readmission rate and improvements in patient care.

In 2017, the hospitalization cost of patients with DM in the USA was $123 billion. Based on a 20% readmission rate, it was estimated that $24.6 billion would be attributed to readmission. A survey conducted by the Agency for Healthcare Research and Quality (AHRQ) found that more than 3.3 million patients were readmitted in the United States after being discharged. The need for readmission indicates that inadequate care was provided to the patient at the time of first admission. Inadequate care poses threat to patients’ life and treatment of readmitted patients leads to increased healthcare costs.

3. Problem Statement:

Machine learning helps in providing more accurate predictions than current practices. Patients facing a high risk of readmission need to be identified at the time of being discharged from the hospital, to facilitate improved treatment to reduce the chances of their readmission. Readmission of patients within 30 days of being discharged (short-term readmission) and those with more than 30 days of readmission has been a widely used metric for studying re-admissions. This project aims to carry out the use cases of “Binomial classification” by combining both types of readmissions for research purposes to check the efficiencies of the different ML algorithms.

4. Project Justification:

The core idea is to provide a comprehensive data solution for readmission problem to facilitate implementation at the healthcare institutions to embark a significant improvement in the in-patient diabetic care.

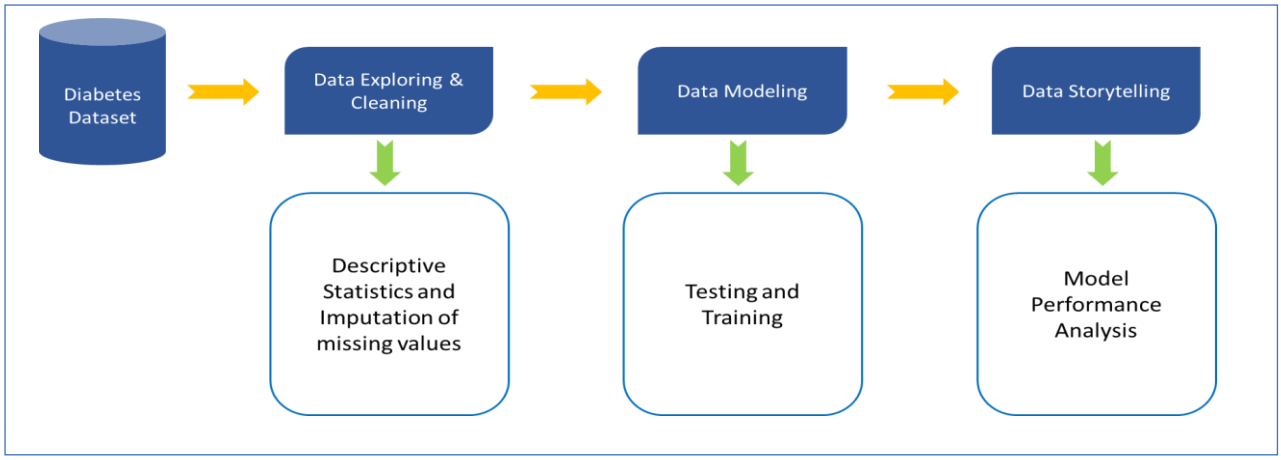
Addressing these critical problems involves several data challenges which are considered throughout the research. The main contribution of this work are:

- Prediction of diabetes patients with high risk of readmission, by modelling bivariate patient medical records using machine learning classifiers.
- Incorporating a conservative prediction model to have higher recall as suited to healthcare institutions.

- Analysis of characteristics of Short-term (within 30 days) and Long-term (after 30 days) re-admissions with different classifiers.
- Cost optimization by implementing the work in real world.

Our goal is to find whether a patient gets readmitted irrespective of the span of 30 days. This results in the highest cost incurred for both the hospital and the patients as reported by funding agencies. Hence our target variable for this dataset is the “Readmitted column” which has 0 and 1 class for never readmitted and re-admitted respectively.

5.Methodology



The Methodology followed by the cross industry standard process for Data Mining (CRISP-DM). CRISP-DM provides the complete blueprint for conducting a Data Mining project. It provides a uniform framework for experience documentation. It breaks down the Life Cycle of a Data Mining project into the following phases:

Phase 1: Business Understanding

Understanding the business requirement is of paramount importance. The primary activity is to find out exactly what business is trying to accomplish. One needs to go deeper into fact finding to outline the issues in the business goals task. After understanding the requirements, the next steps is to lay down every step that the data scientist intends to take until the project is accomplished and results are presented and reviewed.

Phase 2: Data Understanding

The data understanding phase goes hand in hand with the business understanding phase. In this phase, the data will be collected, described, explored and verified. Data Understanding phase is where Exploratory Data Analysis is performed.

Phase 3: Data Preparation

The third phase is Data Preparation which is also called as Data Wrangling or Data Munging Phase. It involves developing the final dataset for modelling. It covers all activities to construct the final dataset from the initial raw data. In this phase Data Cleaning, Imputation and Feature Engineering are performed.

Phase 4: Modelling

The fourth phase is Modelling, which involves selecting the actual modelling technique that needs to be used. In this phase the Modelling technique will be selected, test design will be generated, model will be built. Several models will be built based on tuning iterations until the best model is found.

Phase 5: Evaluation

The fifth phase is Evaluation, where the model will be evaluated. The steps executed to construct the model to ascertain it properly achieves the business objectives.

Phase 6: Deployment

The last and final phase is Deployment. However, for this project, this step will not be executed.

5. Dataset :

This study uses the Health Facts National Database (Cerner Corporation, Kansas City, MO), gathering extensive clinical records across hundreds of hospitals throughout the US. The data subset used for analysis covers 10 years of diabetes patient encounter data (1999 – 2008) among 130 US hospitals with over 100,000 diabetes patients. Moreover, all the encounters used for analysis satisfy five key criteria:

- It is a hospital admission.
- The inpatient was classified as diabetic (at least one of three initial diagnoses included diabetes).
- The length of stay was comprised from 1 to 14 days.
- The inpatient underwent laboratory testing.
- The inpatient received medication during its stay

The dataset has over 49 features split into continuous and categorical ones. The following section would serve as the data dictionary for this given project along with a column for null values.

7. Data Dictionary:

S.no	Feature name	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, new-born, 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 Speciality	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 Outpatient visits	Number of outpatient visits in the year preceding the encounter
14.	Number of lab procedures	Number of lab tests performed during encounter
15.	Number of procedures	Number of procedures (other than lab tests) performed during the encounter
16.	Number of Medications	Number of distinct generic names administered during 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
22.	Number of Diagnoses	Number of diagnoses entered in 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.	Diabetics medication	Indicates if the there was any diabetic medication prescribed. Values: “yes” and “no”
27.	24 features for medication	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

8. Data Exploration (EDA):

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. Since numerous features are present in the dataset, certain features need to be dropped to proceed with feature engineering and exploratory data analysis and the missing value treatment which helps to gain more insights.

8.1 Pre-processing Data Analysis:

The original database contains incomplete, redundant, and noisy information as expected in any real-world data. The features were changed to NaN values for easier processing.

```
df['race'] = df['race'].replace({'?':np.nan})
df['gender'] = df['gender'].replace({'Unknown/Invalid':np.nan})
df['weight'] = df['weight'].replace({'?':np.nan})
df['payer_code'] = df['payer_code'].replace({'?':np.nan})
df['medical_specialty'] = df['medical_specialty'].replace({'?':np.nan})
df[['diag_1','diag_2','diag_3']] = df[['diag_1','diag_2','diag_3']].replace({'?':np.nan})
```

```
total = df.isnull().sum().sort_values(ascending=False)
percentage = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)*100
missing_data = pd.concat([total,percentage], axis=1, keys=['Total', 'Percentage'])
missing_data
```

	Total	Percentage
weight	98569	96.858479
medical_specialty	49949	49.082208
payer_code	40256	39.557416
race	2273	2.233555
diag_3	1423	1.398306
diag_2	358	0.351787
diag_1	21	0.020636
gender	3	0.002948

Weight attribute has the most number of missing values (96.85%) followed by medical_speciality (49%) and payer_code (39.55%) and it is considered not relevant to the outcome and so it was not included in further analysis.

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

The patients expired has been removed from the analysis because it will create bias as they will not be readmitted.

```
df=df[df['discharge_disposition_id']!='Expired']
df=df[df['discharge_disposition_id']!='Neonate discharged']
df=df[df['discharge_disposition_id']!='Hospice']
```

8.2 Encoding Variables:

Since there are more than 800 unique values in each Diagnosis column, we cannot use it, as the computational cost will increase exponentially due to the number of dummy encoded columns needed.

Hence, we categorize them into the major groups based on the ICD 9 standards.

- Circulatory (390–459,785)
- Respiratory (460–519,786)
- Digestive (520–579,787)
- Diabetes (250.xx)
- Injury (800–999)
- Musculoskeletal (710–739)
- Genitourinary (580–629)
- Neoplasms (140–239)
- Other- The codes which are not present in the above list has been classified here.

```
def getCategor(x):
    if 'V' in str(x) or 'E' in str(x):
        return 'Others'

    x = float(x)

    if (x >= 390 and x <= 459) or np.floor(x) == 785:
        return 'Circulatory'
    elif (x >= 460 and x <= 519) or np.floor(x) == 786:
        return 'Respiratory'
    elif (x >= 520 and x <= 579) or np.floor(x) == 787:
        return 'Digestive'
    elif np.floor(x) == 250:
        return 'Diabetes'
    elif x >= 800 and x <= 999:
        return 'Injury'
    elif x >= 710 and x <= 739:
        return 'Musculoskeletal'
    elif (x >= 580 and x <= 629) or np.floor(x) == 788:
        return 'Genitourinary'
    elif x >= 140 and x <= 239 or np.floor(x) in [780, 781, 784] or x >= 790 and x <=799 or x>=240
    and x<=249 or x>=251 and x<=279 or x>=680 and x<=709 or np.floor(x)==782 or x>=1 and x<=139:
        return 'Neoplasms'
    else:
        return 'Others'
```

```
#changing the values into categories
df['Diagnosis1'] = df['diag_1'].apply(lambda x: getCategor(x))
df['Diagnosis2'] = df['diag_2'].apply(lambda x: getCategor(x))
df['Diagnosis3'] = df['diag_3'].apply(lambda x: getCategor(x))
```

Encoding the age range with median values for easier understanding.

```
replacedict = {'[0-10)' : 5,
'[10-20)' : 15,
'[20-30)' : 25,
'[30-40)' : 35,
'[40-50)' : 45,
'[50-60)' : 55,
'[60-70)' : 65,
'[70-80)' : 75,
'[80-90)' : 85,
'[90-100)' : 95}
```

```
df['age'] = df['age'].apply(lambda x : replacedict[x])
```

Replacing the string values of max_glu_serum and A1Cresult by numerical values for the purpose of model building.

```
df.max_glu_serum.replace({'>200':200 , '>300':300 , 'Norm':100 , 'None' : 0} , inplace = True)
```

```
df.A1Cresult.replace({'>7':7 , '>8':8 , 'Norm':5 , 'None':0} , inplace = True)
```

Inconsistencies:

Data inconsistencies compromise data integrity and alter the performance of the algorithm. As a result, the third cleaning step resides in addressing such bias in data. Based on the body of literature this particular set of data has some specific inconsistent features to be addressed.

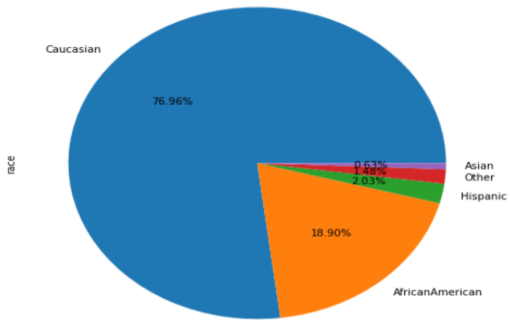
- After doing the above pre-processing, we found that 3 features, named, “examide”, “citoglipton”, “metformin-rosiglitazone” have the same observation (“No”) for every record in the dataset. Such features will, as a result, be dropped from the analysis.
- Discharge Disposition refers to the person's location or status after admission in the healthcare center. Patients who died during their admission have no probability to be readmitted and should be hence excluded from the analysis. Therefore, all records with expired discharge were deleted. Moreover, patients discharged to hospice, referring to terminally ill patients’ care, were also omitted for the same reason.
- We will drop the category ‘new born’ from admission type id because the age corresponding to it give contradictory information. Similarly, we will also drop the category ‘delivery’ from admission source id because the age and gender corresponding to it give contradictory information. Moreover, these observations have counts of up to 10 which is not much.

8.3 Univariate Analysis:

Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable and also describes the data and find patterns that exist within it.

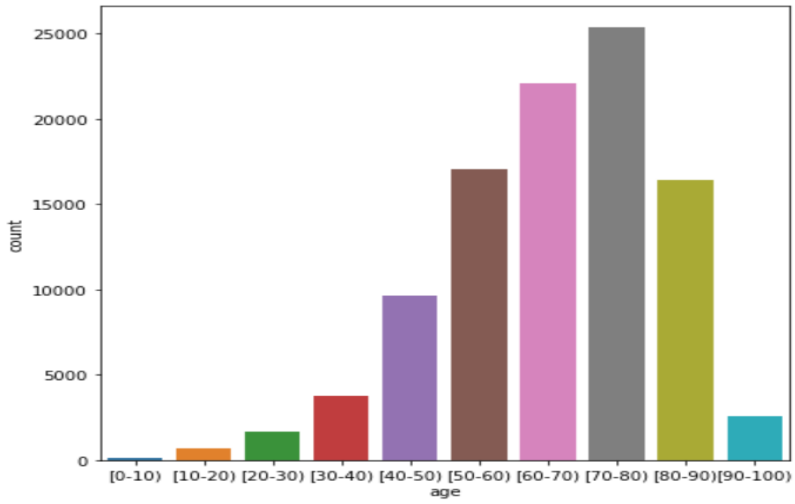
8.3.1 Race:

Majority of the population present in the dataset are Caucasians followed by African-American.



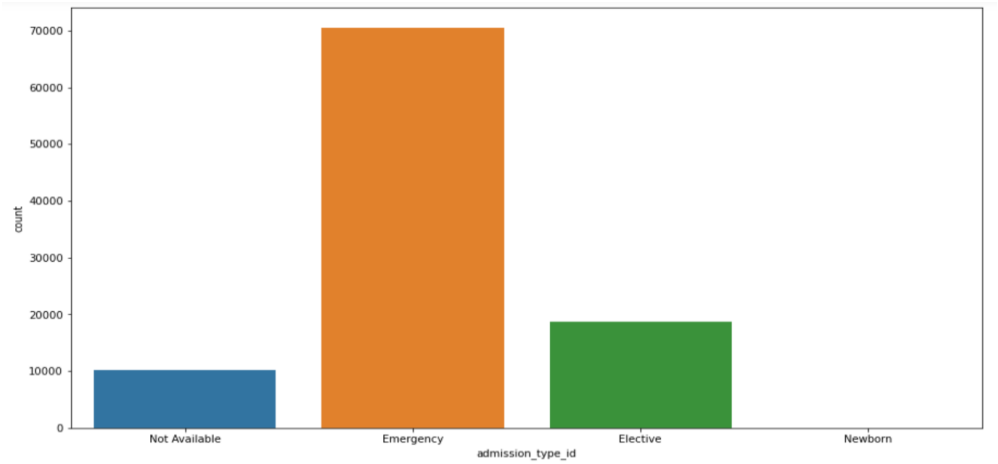
8.3.2 Age:

Patients above the age of 60 are mostly present in the dataset.



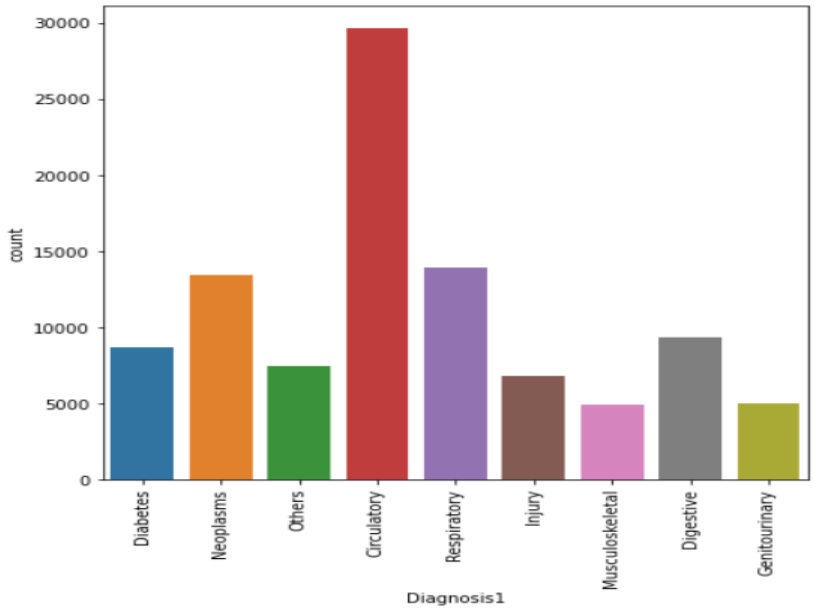
8.3.3 Admission Type Id:

The Admission types have been grouped into 4 categories as mentioned above. The Emergency care patients are the most prevalent ones followed by elective.



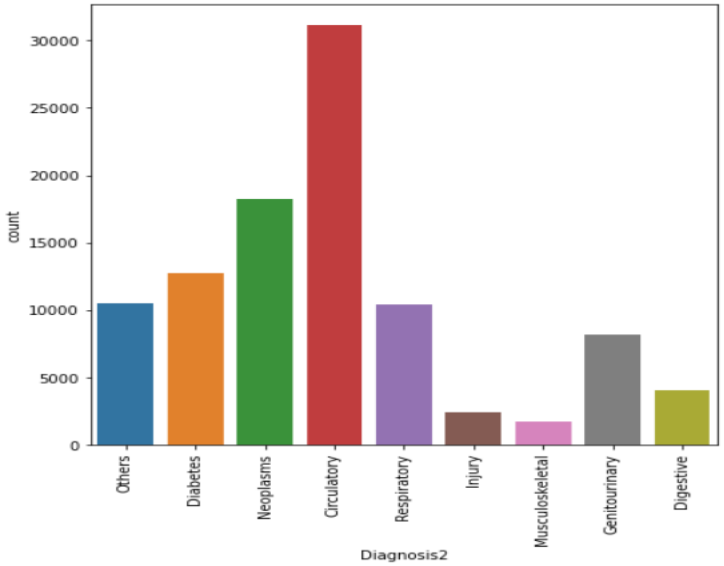
8.3.4 Diagnosis 1:

As we can see, Circulatory diagnosis is performed the most followed by Neoplasms in Diagnosis 1.



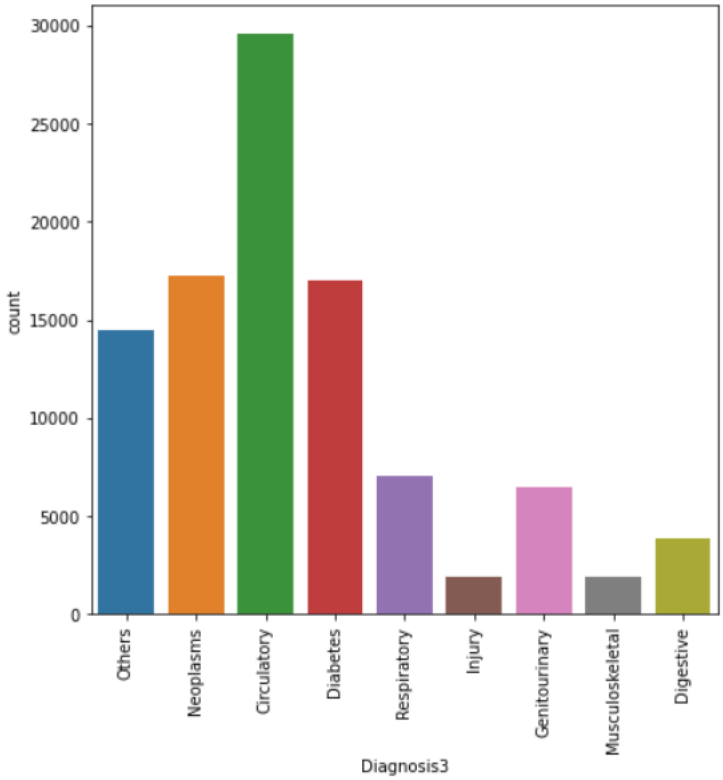
8.3.5 Diagnosis 2:

Circulatory diagnosis is performed the most followed by Neoplasms in Diagnosis 2.



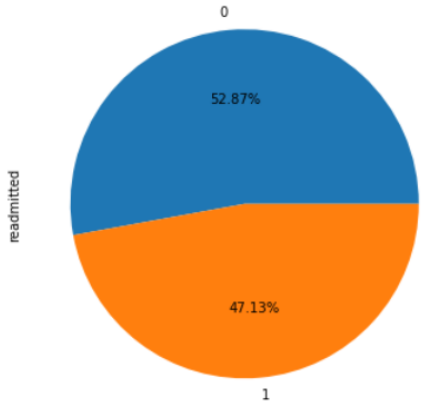
8.3.6 Diagnosis 3:

Circulatory diagnosis is performed the most followed by Neoplasms and Diabetes in Diagnosis 3.



8.3.7 Target Variable:

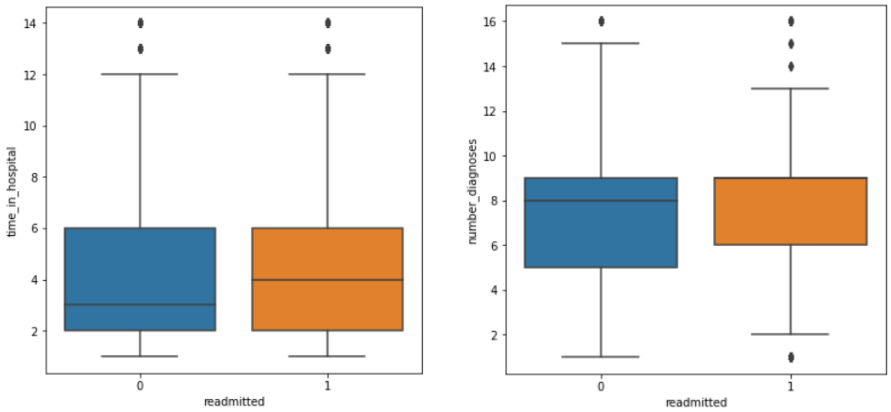
Encoding it such that the target variable satisfies 1 for Yes (re-admitted) and 0 for No (not re-admitted).



According to the pie chart, Patients who are Readmitted >30 days and <30 days percentage is 47.13%.

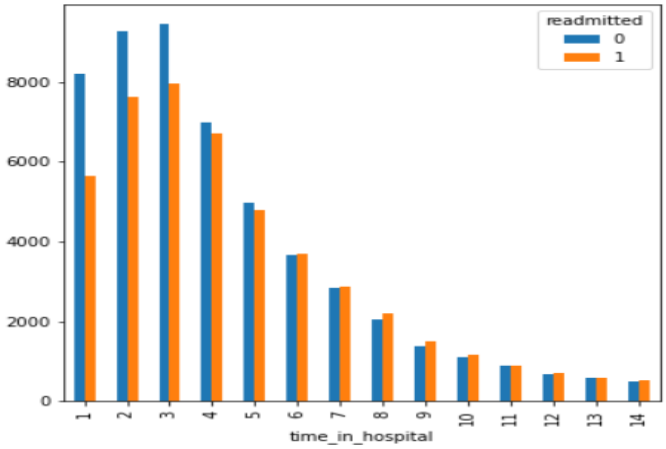
8.4 Bivariate Analysis:

Analysed the impact of the entire continuous variable on the target variable from the box plot and cross tabs. Plotted only a few samples here. There are few outliers in some of the variables. However, all three categories of readmitted categories seem to show similar characteristics.



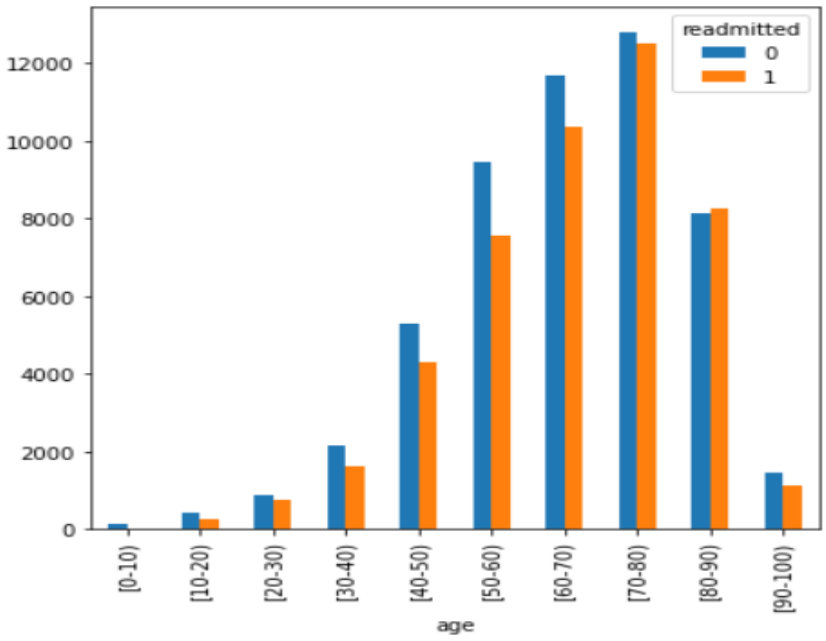
8.4.1 Time in hospital vs Distribution of Readmission:

- The maximum probability of the patients who stayed in the hospital is 3 days.
- The number of patients who are not admitted as well as stayed in the hospital more than 30 days is more.



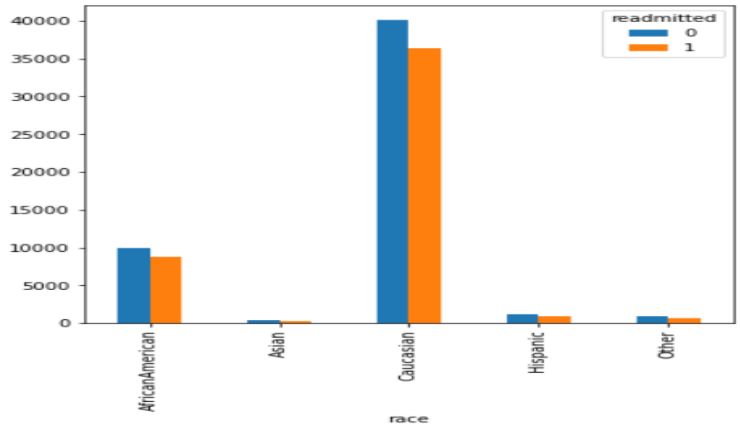
8.4.2 Age of Patient vs Readmission:

Patients above the age of 60 are highly prone to diabetics.



8.4.3 Race vs Readmitted:

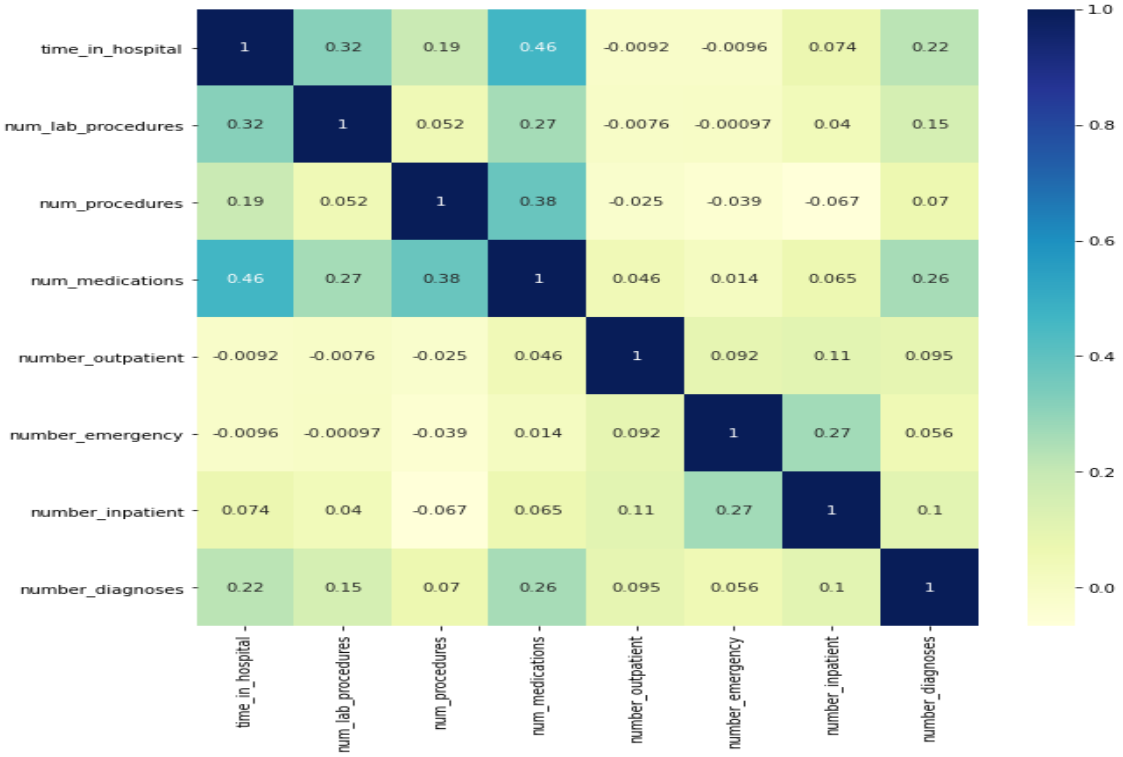
Caucasian race has the highest readmission rate followed by African American.



8.4.4 Heat map:

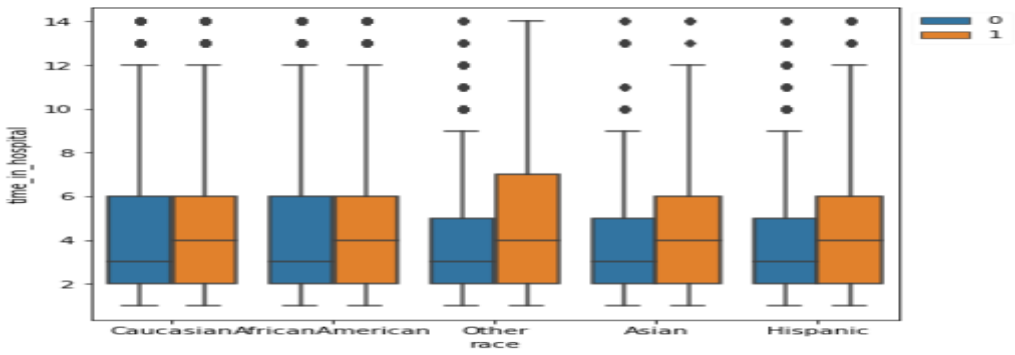
The correlation between the numerical variables is visualized using the heat map.

The correlation of each variable is visualized using the heat map after null values have been imputed successfully.



8.5 Multivariate Analysis:

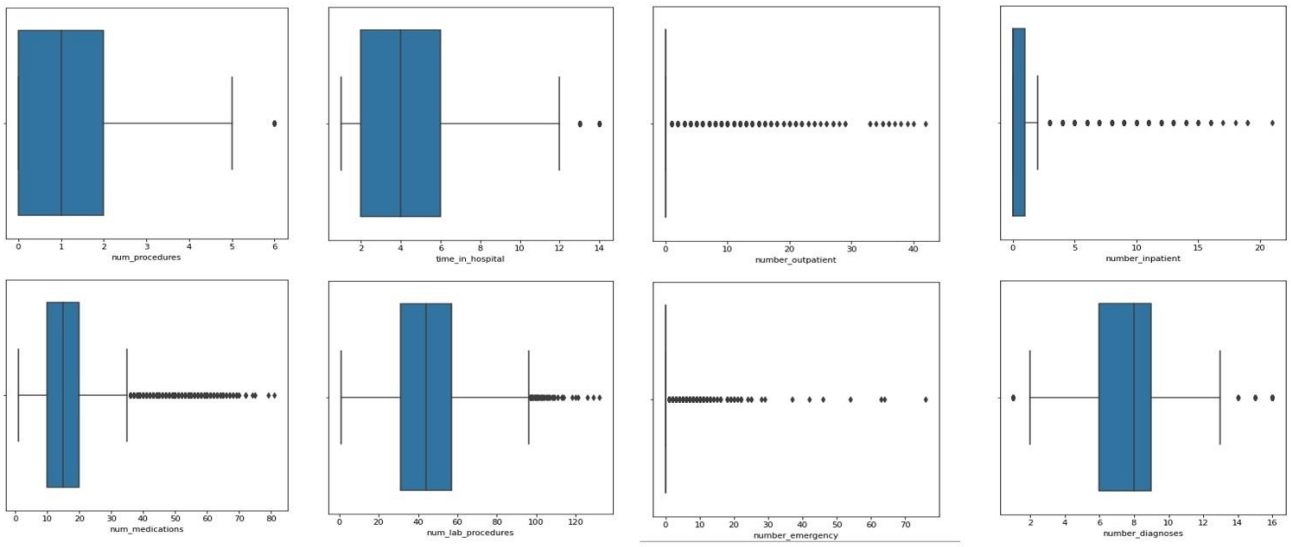
The median values of time in hospital for different races who are admitted is high compared to not admitted.



8.6 Presence of Outliers:

Outliers, being the most extreme observations, may include the sample maximum or sample minimum, or both, depending on whether they are extremely high or low. However, the sample maximum and minimum are not always outliers because they may not be unusually far from other observations.

Box plots show the overall spread of the data while plotting a data point for outliers. This physical point allows their specific values to be easily identified and compared among samples. We generally identify outliers with the help of boxplot, so here box plot shows some of the data points outside the range of the data.



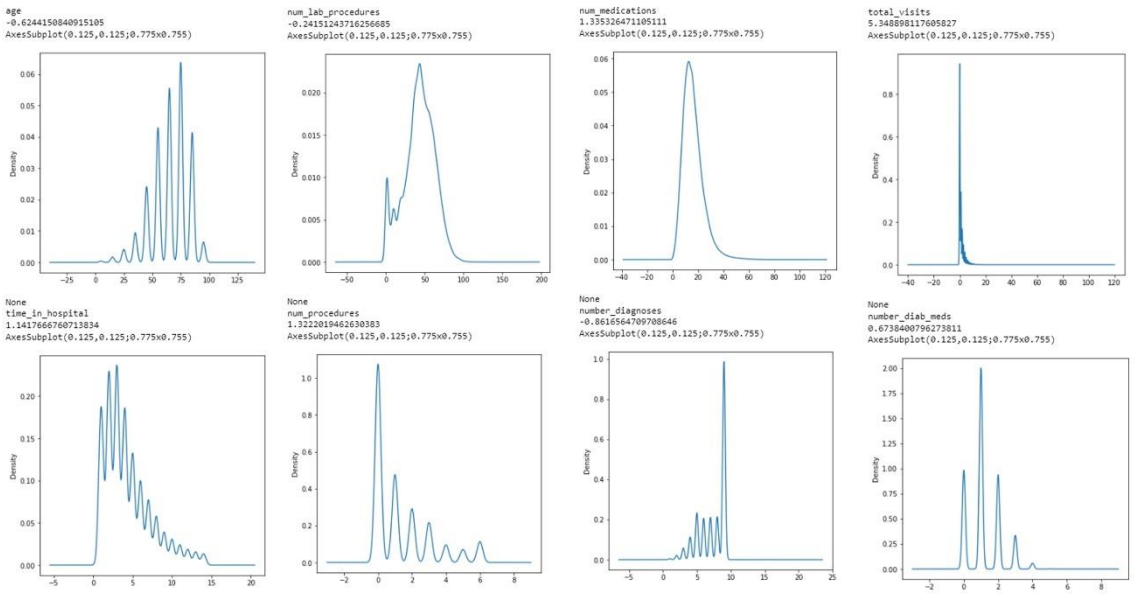
8.7 Distribution of Variables:

An early step in any effort to analyse or model data should be to understand how the variables are distributed. Techniques for distribution visualization can provide quick answers to many important questions. What range do the observations cover? Are they heavily skewed in one direction? Are there significant outliers? etc.

1. **Positively skewed:** Most frequent values are low and tail is towards high values. If **Mode< Median< Mean** then the distribution is positively skewed.

2. **Negatively skewed:** Most frequent values are high and tail is towards low values. If **Mode> Median> Mean** then the distribution is negatively skewed.

   Visualization methods that display frequency, how data spread out over an interval or is grouped. Kernel density estimate (kde) is a quite useful tool for plotting the shape of a distribution, it visualizes the distribution of data over a continuous interval or time period using a continuous probability density curve in one or more dimensions.



9. Data Preprocessing :

9.1 Creating New Features and Dropping Redundant Ones:

'total_visits': 'number_outpatient', 'number_emergency', 'number_inpatient' were combined together by adding them up to get total visits of the patient in the previous year and then dropping the combining features

```
df['total_visits'] = df['number_outpatient'] + df['number_emergency'] + df['number_inpatient']
```

```
df.drop(['number_outpatient' , 'number_emergency','number_inpatient',] , 1 , inplace = True)
```

```
df.change.replace({'No':0,'Ch':1} , inplace = True)
```

We converted all the **medicine** columns into numerical ones by encoding them as:
'No': -2,

'Down': -1,

'Steady': 0,

'Up': 1.

```
meds = ['metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
       'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
       'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
       'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-pioglitazone','metformin-rosiglitazone']
```

```
# The medicines can be encoded as below
for i in meds:
    df[i] = df[i].replace({'No' : -2,
                           'Down' : -1,
                           'Steady' : 0,
                           'Up' : 1})
```

```
for i in meds:
    df[i] = df[i].astype('int64')
```

```
df = df.reset_index(drop = True)
```

'number_diab_meds': This column indicates how many changes has been made to the medicines given to the patient. We dropped the feature 'change' because we created number_changes which is a better feature and captures more information.

```
df['number_diab_meds'] = np.nan
for i in range(len(df)):
    n = 0
    for j in meds:
        if df.loc[i, j] != -2:
            n += 1
    df.loc[i, 'number_diab_meds'] = n
df.number_diab_meds=df.number_diab_meds.astype('int64')
```

'insulin_treatment': Since insulin is the most widely given medicines among all of the medicines, we created a feature which indicates various combinations of insulin administered to the patient, it has following 4 categories:
insulin_only: if only insulin is being given to the patient.

insulin_combo: if insulin was given along with some other medicines.

other_meds: if other medicines were being given but not insulin.

no meds: if no medicine was given to the patient.

```
for i in range(len(df)):
    if df.loc[i, 'insulin'] != -2 and df.loc[i, 'number_diab_meds'] == 1:
        df.loc[i, 'insulin_treatment'] = 'insulin_only'
    elif df.loc[i, 'insulin'] != -2 and df.loc[i, 'number_diab_meds'] > 1:
        df.loc[i, 'insulin_treatment'] = 'insulin_combo'
    elif df.loc[i, 'insulin'] == -2 and df.loc[i, 'number_diab_meds'] == 0:
        df.loc[i, 'insulin_treatment'] = 'no_med'
    else:
        df.loc[i, 'insulin_treatment'] = 'other_meds'
```

9.2 STATISTICAL TEST:

The **statistical significance** indicates that changes in the independent **variables** correlate with shifts in the dependent **variable**. Statistical hypothesis testing is a standard approach to drawing insights from data. It is used to determine whether the result of a data set is statistically significant.

The usual approach to hypothesis testing is to define a question in terms of the variables you are interested in. Then, you can form two opposing hypotheses to answer it.

1. The **null hypothesis** claims there is no statistically significant relationship between the variables.
2. The **alternative hypothesis** claims there is a statistically significant relationship between the variables.

```
#Statistical Tests (Chi Square and Anova)
from scipy import stats
p_val = []
sig = []
for i in df.columns:
    if i in num_data.columns:
        stat, p = stats.f_oneway(df[df['readmitted'] == 0][i], df[df['readmitted'] == 1][i])
    else:
        ct = pd.crosstab(df[i], df['readmitted'])
        stat, p, dof, exp = stats.chi2_contingency(ct)
    p_val.append(p)
    if p < 0.05:
        sig.append('Significant')
    else:
        sig.append("Insignificant")
stats_df = pd.DataFrame({"columns" : df.columns, "p_value" : p_val, "significance" : sig})

stats_df.sort_values(by='p_value',ascending = True)
```

When performing a statistical test, a *p*-value helps us determine the significance of our results in relation to the null hypothesis.

- A *p*-value less than 0.05 (typically ≤ 0.05) is statistically significant. It indicates strong evidence against the null hypothesis, as there is less than a 5% probability the null is correct (and the results are random). Therefore, we reject the null hypothesis, and accept the alternative hypothesis.
  However, this does not mean that there is a 95% probability that the alternative hypothesis is true. The *p*-value is conditional upon the null hypothesis being true is unrelated to the truth or falsity of the alternative hypothesis.
- A *p*-value higher than 0.05 (> 0.05) is not statistically significant and indicates strong evidence for the null hypothesis. This means we retain the null hypothesis and reject the alternative hypothesis.

Significant Values:

	columns	p_value	significance
40	total_visits	0.000000e+00	Significant
36	readmitted	0.000000e+00	Significant
10	number_diagnoses	0.000000e+00	Significant
5	admission_source_id	1.583463e-228	Significant
28	insulin	1.612323e-110	Significant
4	discharge_disposition_id	4.613055e-105	Significant
42	insulin_treatment	3.220842e-92	Significant
35	diabetesMed	7.155342e-75	Significant
3	admission_type_id	8.339376e-75	Significant
6	time_in_hospital	2.374671e-74	Significant
9	num_medications	1.829657e-62	Significant
38	Diagnosis2	2.775831e-58	Significant
37	Diagnosis1	5.086492e-57	Significant
7	num_lab_procedures	2.484424e-54	Significant
39	Diagnosis3	3.123961e-51	Significant
2	age	8.354521e-43	Significant
34	change	1.492819e-41	Significant
8	num_procedures	2.131487e-40	Significant
13	metformin	2.480043e-28	Significant
0	race	8.019063e-18	Significant
12	A1Cresult	2.226558e-16	Significant
11	max_glu_serum	6.582001e-16	Significant
41	number_diab_meds	5.079661e-13	Significant
14	repaglinide	3.047512e-10	Significant
1	gender	4.851099e-08	Significant
19	glipizide	5.881203e-07	Significant
23	rosiglitazone	1.156722e-04	Significant
24	acarbose	1.404204e-04	Significant
22	pioglitazone	1.953247e-02	Significant
20	glyburide	4.215530e-02	Significant

Insignificant Values:

25	miglitol	6.860895e-02	Insignificant
29	glyburide-metformin	7.354352e-02	Insignificant
27	tolazamide	9.025692e-02	Insignificant
16	chlorpropamide	2.025557e-01	Insignificant
17	glimepiride	4.029164e-01	Insignificant
30	glipizide-metformin	4.453986e-01	Insignificant
32	metformin-rosiglitazone	5.307135e-01	Insignificant
21	tolbutamide	5.413807e-01	Insignificant
15	nateglinide	5.763337e-01	Insignificant
26	troglitazone	9.206198e-01	Insignificant
33	metformin-pioglitazone	9.541197e-01	Insignificant
18	acetohexamide	9.541197e-01	Insignificant
31	glimepiride-pioglitazone	9.541197e-01	Insignificant

```
insignificant_meds = ['glimepiride','nateglinide','chlorpropamide','acetohexamide','tolbutamide','acarbose','miglitol',
                      'troglitazone','tolazamide','glyburide-metformin','glipizide-metformin',
                      'glimepiride-pioglitazone','metformin-pioglitazone','glyburide','rosiglitazone',
                      'metformin-rosiglitazone']
```

```
df.drop(['glimepiride','nateglinide','chlorpropamide','acetohexamide','tolbutamide','acarbose','miglitol',
                      'troglitazone','tolazamide','glyburide-metformin','glipizide-metformin',
                      'glimepiride-pioglitazone','metformin-pioglitazone','glyburide','rosiglitazone',
        'metformin-rosiglitazone'],1,inplace=True)
```

10. FEATURE ENGINEERING:

10.1 Transformation:

If a measurement variable does not fit a normal distribution or has greatly different standard deviations in different groups, we should try data transformation so that the variables are as close to the normal distribution as possible.

```
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer()
num_data_pt = pt.fit_transform(num_data)
num_data_pt=pd.DataFrame(num_data_pt,columns=num_data.columns)
num_data_pt.skew()
```

10.2 Scaling:

Feature scaling (also known as data normalization) is the method used to standardize the range of features of data. Since the range of values of data may vary widely, it becomes a necessary step while using machine learning algorithms. In this method, we convert variables with different scales of measurements into a single scale.

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
scaledXtrain = sc.fit(X_train)
scaledXtrain = sc.transform(X_train)
scaledXtest = sc.transform(X_test)
```

10.3 Feature Selection

The feature that we are selecting for base model building are as follows.

['gender', 'age', 'time_in_hospital', 'num_lab_procedures',
'num_procedures', 'num_medications', 'number_diagnoses', 'metformin',
'repaglinide', 'glipizide', 'pioglitazone', 'insulin', 'change',
'diabetesMed', 'readmitted', 'total_visits', 'number_diab_meds',
'race_Asian', 'race_Caucasian', 'race_Hispanic', 'race_Other',
'discharge_disposition_id_Discharged to home with home health service',
'discharge_disposition_id_Left AMA',
'discharge_disposition_id_Not Available',

'discharge_disposition_id_Still patient/referred to this institution',
'discharge_disposition_id_Transferred to another medical facility',
'admission_source_id_Not Available', 'admission_source_id_Referral',
'admission_source_id_Transferred from another health care facility',
'max_glu_serum_100', 'max_glu_serum_200', 'max_glu_serum_300',
'A1Cresult_5', 'A1Cresult_7', 'A1Cresult_8', 'Diagnosis2_Diabetes',
'Diagnosis2_Digestive', 'Diagnosis2_Genitourinary', 'Diagnosis2_Injury',
'Diagnosis2_Musculoskeletal', 'Diagnosis2_Neoplasms',
'Diagnosis2_Others', 'Diagnosis2_Respiratory', 'Diagnosis1_Diabetes',
'Diagnosis1_Digestive', 'Diagnosis1_Genitourinary', 'Diagnosis1_Injury',
'Diagnosis1_Musculoskeletal', 'Diagnosis1_Neoplasms',
'Diagnosis1_Others', 'Diagnosis1_Respiratory', 'Diagnosis3_Diabetes',
'Diagnosis3_Digestive', 'Diagnosis3_Genitourinary', 'Diagnosis3_Injury',
'Diagnosis3_Musculoskeletal', 'Diagnosis3_Neoplasms',
'Diagnosis3_Others', 'Diagnosis3_Respiratory',
'insulin_treatment_insulin_only', 'insulin_treatment_no_med',
'insulin_treatment_other_meds']

11. Model Building :

11.1 Train and Test split:

The train_test_split function is for splitting a single dataset for two different purposes: training and testing. The training subset is for building your model and the testing subset is for using the model on unknown data to evaluate the performance of the model.

Parameters:

X, y: The first parameter is the dataset you're selecting to use.

test_size: This parameter specifies the size of the testing dataset. The default state suits the training size. It will be set to 0.25 if the training size is set to default.

random_state: Controls the shuffling applied to the data before applying the split. The default mode performs a random split using np.random. Alternatively, you can add an integer using an exact number.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,random_state=1)
```

11.2 Base Model:

After cleaning the initial dataset and post one hot encoding the variables we have 99323 rows and 62 columns. So, we proceeded to build a few base models using Logistic Regression, Decision Tree, Random Forest, KNN, AdaBoost, Gradient Boost, XGBoost, LGBoost, Catboost, Naïve Bayes on this data without scaling or transformation or hyperparameter tuning until the model performance scores are in acceptable range. Below is the data frame consisting of all the scores.

In this section, we will first compare the performance of the following 10 machine learning models using default hyperparameters:

- Logistic regression
- Decision Tree Classifier
- Random Forest Classifier
- KNeighbors Classifier
- AdaBoost Classifier
- Gradient Boost Classifier
- XGB Classifier
- LGBM Classifier
- Cat Boost Classifier
- GaussianNB

1) Logistic Regression: Logistic regression is the classification counterpart to linear regression. Predictions are mapped to be between 0 and 1 through the logistic function, which means that predictions can be interpreted as class probabilities. The models themselves are still "linear," so they work well when your classes are linearly separable (i.e. they can be separated by a single decision surface.

The logistic regression method assumes that:

- The outcome is a binary or dichotomous variable like yes vs no, positive vs negative or 1

vs 0.

- There is a linear relationship between the logit of the outcome and each predictor variables. (Recall that the logit function is logit(p) = log(p/(1-p)), where p is the probabilities of the outcome.)
- There are no influential values (extreme values or outliers) in the continuous predictors.
- There is no high intercorrelations (i.e. multicollinearity) among the predictors.

Strengths: Outputs have a nice probabilistic interpretation, and the algorithm can also be regularized by penalizing coefficients with a tunable penalty strength to avoid overfitting. Logistic models can be updated easily with new data using stochastic gradient descent.

Weaknesses: Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

2) Decision Trees: A Decision Tree is a simple representation for classifications and regression. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. In Decision Tree as we have no probabilistic model, but just binary split, we don't need to make any assumption at all.

Strengths: As with regression, classification tree ensembles also perform very well in practice. They are robust to outliers, scalable, and able to naturally model non-linear decision boundaries thanks to their hierarchical structure.

Weaknesses: Unconstrained, individual trees are prone to overfitting, but this can be alleviated by ensemble methods.

3) Random Forests: Random Forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. By considering more than one decision tree and then doing a majority voting, random forests helped in being more robust predictive representations than trees as in the previous case. It has no model underneath, and the only assumption that it relies is that sampling is representative. But this is usually a common assumption.

Strengths: Random forest can solve both type of problems that is classification and regression and does a decent estimation at both fronts. It can handle thousands of input variables and identity most significant variables so it is considered as one of the dimensionality reduction method. Further, the model outputs importance of variable, which can be a very handy feature. It has an effective method for estimating missing data and maintains accuracy when large

proportion of the data are missing. It has methods for balancing errors in data sets where classes are imbalanced.

Weaknesses: It surely does a good job at classification but not as for regression problem as it does not give precise continuous nature prediction. In case of regression, it doesn't predict beyond the range in the training data, and that they may over fit data sets that are particularly noisy. Random forest can feel like a black box approach as we have very little control on what the model does. You can at best try different parameters and random seeds.

4) LightGBM Classifier: Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.

Strengths:

- **Faster training speed and higher efficiency**: Light GBM use histogram-based algorithm i.e. it buckets continuous feature values into discrete bins which fasten the training procedure.
- **Lower memory usage:** Replaces continuous values to discrete bins which result in lower memory usage.
- **Better accuracy than any other boosting algorithm:** It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
- **Compatibility with Large Datasets:** It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.

Weaknesses: It can overfit if there are less than 10,000 observations in the dataset.

5) XGBoost Classifier: XGBoost stands for extreme Gradient Boosting. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

Strengths: XGBoost is fast when compared to other implementations of gradient boosting. It dominates structured or tabular datasets on classification and regression predictive modeling problems.

Weaknesses: The implementation of XGBoost is relatively. Therefore, it lacks scalability. It also uses a lot of memory.

6) **K nearest neighbors (KNN)**

KNN is one the simplest machine learning models. For a given sample point, the model looks at the k closest datapoints and determines the probability by counting the number of positive labels divided by K. This model is easy to implement and understand, but comes at the disadvantage of being sensitivity to K and takes a long time to evaluate if the number of trained samples is large.

7) **Naive Bayes**

Naive Bayes is another model occasionally used in machine learning. In Naive Bayes, we utilize Bayes rule to calculate the probabilities. The "naive" part of this model is that it assumes all the features are independent (which is generally not the case). This works well for natural language processing models, but let's try it out here anyways.

11.2.1 Evaluation of Model :

Evaluating machine learning algorithms is an essential part of any project. After building a predictive classification model, we need to evaluate the performance of the model, that is how good the model is in predicting the outcome of new observations test data that have been not used to train the model.

In other words, we need to estimate the model prediction accuracy and prediction errors using a new test data set. Because we know the actual outcome of observations in the test data set, the performance of the predictive model can be assessed by comparing the predicted outcome values against the known outcome values.

Evaluation Metrics

The commonly used metrics and methods for assessing the performance of predictive classification models, including:

Confusion Matrix: Confusion Matrix as the name suggests gives us a 2x2 matrix as output and describes the complete performance of the model. Here are 4 important terms:

- **True Positives**: The cases in which we predicted YES and the actual output was also YES.
- **True Negatives:** The cases in which we predicted NO and the actual output was NO.
- **False Positives**: The cases in which we predicted YES and the actual output was NO.
- **False Negatives**: The cases in which we predicted NO and the actual output was YES.

Accuracy: It represents the proportion of correctly classified observations. Accuracy for the matrix can be calculated by taking average of the values lying across the "main diagonal".

Classification Report: The classification report visualizer displays the precision, recall, F1, and support scores for the model. The classification report shows a representation of the main classification metrics on a per-class basis. This gives a deeper intuition of the classifier behavior over global accuracy which can mask functional weaknesses in one class of a multiclass problem.

Precision: Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives. Said another way, "for all instances classified positive, what percent was correct?"

Recall: Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. Said another way, "for all instances that were actually positive, what percent was classified correctly?"

F1 score: The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

ROC curve: It is a graphical summary of the overall performance of the model, showing the proportion of true positives and false positives at all possible values of probability cut-off. The Area Under the Curve (**AUC**) summarizes the overall performance of the classifier.

Metrics of importance in our project

The recall is the measure of our model correctly identifying True Positives. Mathematically:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positive + False Negative}}$$

Recall also gives a measure of how accurately our model is able to identify the relevant data. We refer to it as Sensitivity or True Positive Rate. Higher sensitivity (recall) is more desirable for hospitals because it is more crucial to correctly identify "high risk" patients who are likely to be readmitted than identifying "low risk" patients.

	Model	Precision_Train	Recall_Train	F1_Train	Accuracy_Train	ROC_AUC_Train
0	LR	0.616168	0.533529	0.571878	0.624601	0.666814
1	DT	1.000000	0.999969	0.999985	0.999986	1.000000
2	RF	1.000000	0.999969	0.999985	0.999986	1.000000
3	KNN	0.710050	0.681603	0.695535	0.719573	0.789068
4	ADB	0.623657	0.515257	0.564299	0.626082	0.669526
5	GB	0.630141	0.528908	0.575104	0.632727	0.681864
6	XGB	0.700399	0.650751	0.674663	0.705060	0.780840
7	LGB	0.653106	0.577847	0.613176	0.657380	0.717750
8	CB	0.615845	0.529122	0.569199	0.623608	0.664370
9	NB	0.557727	0.607811	0.581693	0.589190	0.617393

	Model_Test	Precision_Test	Recall_Test	F1_Test	Accuracy_Test	ROC_AUC_Test
0	LR	0.622083	0.535654	0.575642	0.625331	0.665916
1	DT	0.528666	0.527731	0.528198	0.552740	0.551523
2	RF	0.617847	0.539261	0.575886	0.623184	0.667138
3	KNN	0.541289	0.510540	0.525465	0.562540	0.585155
4	ADB	0.627613	0.516129	0.566438	0.625164	0.667981
5	GB	0.629873	0.528014	0.574463	0.628889	0.675674
6	XGB	0.616733	0.569963	0.592426	0.627949	0.674233
7	LGB	0.627271	0.554542	0.588668	0.632346	0.679567
8	CB	0.618597	0.527094	0.569191	0.621472	0.663969
9	NB	0.554927	0.597128	0.575255	0.581669	0.609713

11.3 Hyper Parameter Tuning

Hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. Hyperparameter settings could have a big impact on the prediction accuracy of the trained model. Optimal hyperparameter settings often differ for different datasets and different models. Therefore they should be tuned for each dataset and model. Since the training process doesn't set the hyperparameters, there needs to be a meta process that tunes the hyperparameters.

Hyperparameter Tuning Algorithms:

1. Grid Search: Grid search, true to its name, picks out a grid of hyperparameter values, evaluates every one of them, and returns the winner.

2. Random Search: Random search is a slight variation on grid search. Instead of searching over the entire grid, random search only evaluates a random sample of points on the grid. This makes random search a lot cheaper than grid search.

XGBoost Classifier:

```
#XGB
```

```
tuned_paramaters = [{'n_estimators': [100, 120, 150],
                     'learning_rate': [0.1, 0.01, 0.001, 0.15, 0.015],
                     'gamma': [2, 3, 4, 5, 6],
                     'max_depth': [2, 3, 4, 5, 6]}]


xgb_model_classifier = XGBClassifier(random_state = 8, n_jobs=-1)

xgb_grid = GridSearchCV(estimator = xgb_model_classifier,
                        param_grid = tuned_paramaters,
                        cv = 5,
                        n_jobs=-1)

# fit the model on X_train and y_train using fit()
xgb_model=xgb_grid.fit(X_train, y_train)

# get the best parameters
print('Best parameters for Extreme Gradient Boositng Classifier: ', xgb_model.best_params_, '\n')
```

```
Best parameters for Extreme Gradient Boositng Classifier:  {'gamma': 4, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimator
s': 120}
```

```
XGB_tuned_model = XGBClassifier( n_estimators = xgb_model.best_params_['n_estimators'],
                                        learning_rate = xgb_model.best_params_['learning_rate'],
                                        gamma = xgb_model.best_params_['gamma'],
                                        max_depth = xgb_model.best_params_['max_depth'],
                                        random_state = 10)
scores = cross_val_score(estimator = XGB_tuned_model,
                         X = X_train,
                         y = y_train,
                         cv = 5,
                         scoring = 'recall')
print("Mean Recall score after 5 fold cross validation: ", round(scores.mean(), 2))
```

```
Mean Recall score after 5 fold cross validation:  0.55
Train Accuracy Score: 0.6646002934154129
Train Confusion Matrix:
 [[26988  9865]
 [13454 19219]]

Train ClassificationReport :
                precision    recall  f1-score   support

           0       0.67      0.73      0.70     36853
           1       0.66      0.59      0.62     32673

    accuracy                           0.66     69526
   macro avg       0.66      0.66      0.66     69526
weighted avg       0.66      0.66      0.66     69526


Train F1 Score: 0.6224071765144031
Train Precision Score: 0.6608100673910053
Train Recall Score: 0.5882226915189912
Train ROC_AUC Score: 0.7275968723441246

Test Accuracy Score: 0.6309695606940297
Test Confusion Matrix:
 [[10946  4715]
 [ 6281  7855]]

Test ClassificationReport :
                precision    recall  f1-score   support

           0       0.64      0.70      0.67     15661
           1       0.62      0.56      0.59     14136

    accuracy                           0.63     29797
   macro avg       0.63      0.63      0.63     29797
weighted avg       0.63      0.63      0.63     29797


Test F1 Score: 0.5882573204523328
Test Precision Score: 0.6249005568814638
Test Recall Score: 0.5556734578381437
Test ROC_AUC Score: 0.6815796777738522
```

LGBM Classifier:

```
tuned_paramaters = [{'n_estimators': [100, 120, 150],
                     'learning_rate': [0.001, 0.01, 0.1],
                     'max_depth': [2, 3, 4, 5, 6]}]

lgb_model_classifier = LGBMClassifier(random_state = 8, n_jobs=-1)

lgb_grid = GridSearchCV(estimator = lgb_model_classifier,
                        param_grid = tuned_paramaters,
                        cv = 5,
                        n_jobs=-1)

# fit the model on X_train and y_train using fit()
lgb_model=lgb_grid.fit(X_train, y_train)

# get the best parameters
print('Best parameters for Light Gradient Boositng Classifier: ', lgb_model.best_params_, '\n')
```

```
Best parameters for Light Gradient Boositng Classifier:  {'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 120}
```

```
lgb_model.best_score_
```

```
0.6318356708806595
```

```
LGB_tuned_model = LGBMClassifier(n_estimators = lgb_model.best_params_['n_estimators'],
                                         learning_rate = lgb_model.best_params_['learning_rate'],
                                         max_depth = lgb_model.best_params_['max_depth'],
                                         random_state = 10)
scores = cross_val_score(estimator = LGB_tuned_model,
                         X = X_train,
                         y = y_train,
                         cv = 5,
                         scoring = 'recall')
print("Mean Recall score after 5 fold cross validation: ", round(scores.mean(), 2))
```

```
Mean Recall score after 5 fold cross validation:  0.55
```

```
Train Accuracy Score: 0.6579552972988523
Train Confusion Matrix:
 [[24361 12492]
 [11289 21384]]

Train ClassificationReport :
              precision    recall  f1-score   support

           0       0.68      0.66      0.67     36853
           1       0.63      0.65      0.64     32673

    accuracy                           0.66     69526
   macro avg       0.66      0.66      0.66     69526
weighted avg       0.66      0.66      0.66     69526


Train F1 Score: 0.6426542848126944
Train Precision Score: 0.6312433581296493
Train Recall Score: 0.6544853548801763
Train ROC_AUC Score: 0.7190456851401197

Test Accuracy Score: 0.6279826828204181
Test Confusion Matrix:
 [[9820 5841]
 [5244 8892]]

Test ClassificationReport :
              precision    recall  f1-score   support

           0       0.65      0.63      0.64     15661
           1       0.60      0.63      0.62     14136

    accuracy                           0.63     29797
   macro avg       0.63      0.63      0.63     29797
weighted avg       0.63      0.63      0.63     29797


Test F1 Score: 0.6160241089057465
Test Precision Score: 0.6035430665852168
Test Recall Score: 0.6290322580645161
Test ROC_AUC Score: 0.6797108403946419
```

Naïve Bayes:

```
#NB
```

```
param = {'var_smoothing': np.logspace(0,-9, num=100)}
nb_classifier = GaussianNB()
nb_grid = RandomizedSearchCV(estimator = nb_classifier,param_distributions=param,cv = 5,scoring='recall')

# use fit() to fit the model on the train set
nb_model = nb_grid.fit(X_train, y_train)

# get the best parameters
print('Best parameters for random forest Classifier RCV: ', nb_model.best_params_, '\n')
```

```
Best parameters for random forest Classifier RCV:  {'var_smoothing': 8.111308307896873e-06}
```

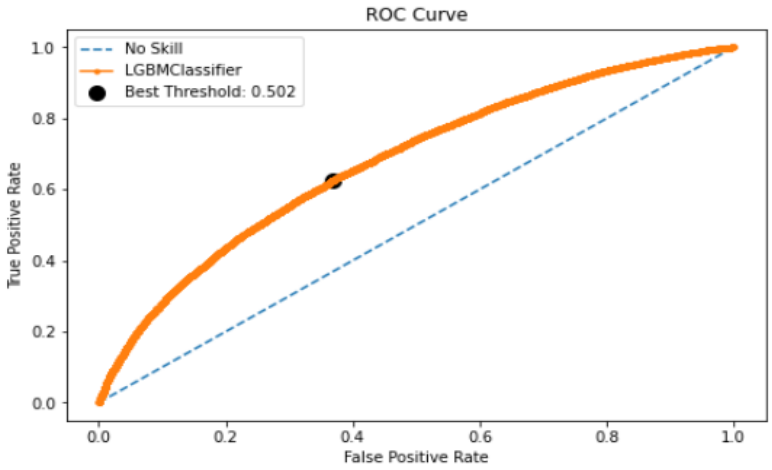
```
nb_model1 = GaussianNB(var_smoothing = nb_model.best_params_['var_smoothing'])
scores = cross_val_score(estimator = nb_model1,
                         X = X_train,
                         y = y_train,
                         cv = 5,
                         scoring = 'recall')
print("Mean Recall score after 5 fold cross validation: ", round(scores.mean(), 2))
```

```
Mean Recall score after 5 fold cross validation:  0.62

Train Accuracy Score: 0.5892184218853379
Train Confusion Matrix:
 [[21107 15746]
 [12814 19859]]

Train ClassificationReport :
               precision    recall  f1-score   support

           0       0.62      0.57      0.60     36853
           1       0.56      0.61      0.58     32673

    accuracy                           0.59     69526
   macro avg       0.59      0.59      0.59     69526
weighted avg       0.59      0.59      0.59     69526


Train F1 Score: 0.5817100676645479
Train Precision Score: 0.5577587417497543
Train Recall Score: 0.6078107305726441
Train ROC_AUC Score: 0.6173923695578986

Test Accuracy Score: 0.5817028559922139
Test Confusion Matrix:
 [[8892 6769]
 [5695 8441]]

Test ClassificationReport :
               precision    recall  f1-score   support

           0       0.61      0.57      0.59     15661
           1       0.55      0.60      0.58     14136

    accuracy                           0.58     29797
   macro avg       0.58      0.58      0.58     29797
weighted avg       0.58      0.58      0.58     29797


Test F1 Score: 0.5752743133646834
Test Precision Score: 0.5549638395792242
Test Recall Score: 0.5971279003961517
Test ROC_AUC Score: 0.609712729511274
```

12. Final Model:

As we can see from the above results, LGBM and Naïve Bayes are giving better results after hyperparameter tuning. But the training and prediction time of Naïve Bayes is far more than that of LGBM. So Naïve Bayes is not scalable and not suitable for production because it will increase costs and time while giving similar results as LGBM which takes only 3-4 seconds to train on this large dataset thus saving costs and time while giving the best results.

ROC Curve:



ROC Curve gives an idea about how well the model performs across all thresholds. Best threshold is selected for which the area under the ROC Curve is maximum, giving the best of both Sensitivity and Specificity i.e. maximum true positive rate while minimizing the false positive rate.

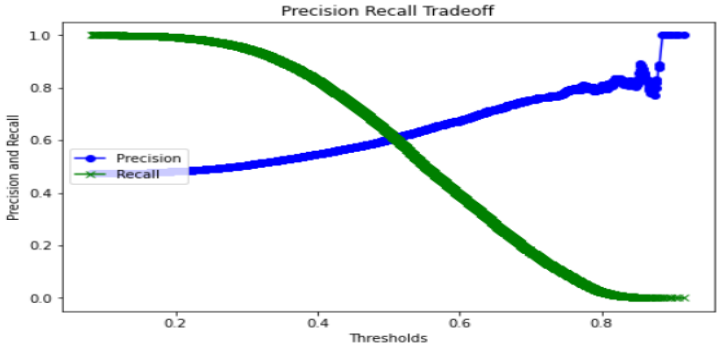
After tuning the threshold to the best one according to the ROC Curve i.e. 0.502, following are the results of the final Model chosen to put in production:

```
Train Accuracy Score: 0.6579552972988523
Train Confusion Matrix:
 [[24361 12492]
 [11289 21384]]

Train ClassificationReport :
                precision    recall  f1-score   support

           0       0.68      0.66      0.67     36853
           1       0.63      0.65      0.64     32673

    accuracy                           0.66     69526
   macro avg       0.66      0.66      0.66     69526
weighted avg       0.66      0.66      0.66     69526


Train F1 Score: 0.6426542848126944
Train Precision Score: 0.6312433581296493
Train Recall Score: 0.6544853548801763
Train ROC_AUC Score: 0.7190456851401197

Test Accuracy Score: 0.6279826828204181
Test Confusion Matrix:
 [[9820 5841]
 [5244 8892]]

Test ClassificationReport :
                precision    recall  f1-score   support

           0       0.65      0.63      0.64     15661
           1       0.60      0.63      0.62     14136

    accuracy                           0.63     29797
   macro avg       0.63      0.63      0.63     29797
weighted avg       0.63      0.63      0.63     29797


Test F1 Score: 0.6160241089057465
Test Precision Score: 0.6035430665852168
Test Recall Score: 0.6290322580645161
Test ROC_AUC Score: 0.6797108403946419
```

Confusion Matrix:

9820	**5841**
5244	8892

This is the confusion matrix given by the final model on the test dataset. As we can see below in the Precision Recall Tradeoff graph, as we increase the Recall, the precision decreases which means that if we want to reduce the number of False Negatives our False Positives will increase. Given that our primary metric is Recall, we have chosen the threshold that's giving us good recall while manageable precision.



Confidence Interval for Recall:

It is important to both present the expected skill of a machine learning model as well as confidence intervals for that model skill.

Confidence intervals provide a range of model skills and a likelihood that the model skill will fall between the ranges when making predictions on new data. For example, a 95% likelihood of classification accuracy between 60% and 70%.

A robust way to calculate confidence intervals for machine learning algorithms is to use the bootstrap. This is a general technique for estimating statistics that can be used to calculate empirical confidence intervals, regardless of the distribution of skill scores (e.g. non-Gaussian).

Since our primary metric is Recall, we will find the confidence interval for that metric. At 95%

Level of Significance the confidence interval for the Recall is: (59 %, 63 %).

This means that we are 95% confident that the recall of the model will lie between 59 % to 63 %.

13. Model Interpretability:

A machine learning algorithm's interpretability refers to how easy it is for humans to understand the processes it uses to arrive at its outcomes. Some models, like logistic regression, are considered to be fairly straightforward and therefore highly interpretable, but as you add features or use more complicated machine learning models such as deep learning, interpretability gets more and more difficult.

Why is Model Interpretability Important?

When using an algorithm's outcomes to make high-stakes decisions, it's important to know which features the model did and did not take into account. Additionally, if a model isn't highly interpretable, the business might not be legally permitted to use its insights to make changes to processes. In heavily regulated industries like banking, insurance, and healthcare, it is important to be able to understand the factors that contribute to likely outcomes in order to comply with regulation and industry best practices.

Some models have inbuilt properties that provide these sorts of explanations. These are typically referred to as white box models, and examples include linear regression (model coefficients), logistic regression (model coefficients) and decision trees (feature importance). Due to their complexity, other models – such as Random Forests, Gradient Boosted Trees, SVMs, Neural Networks, etc. – do not have straightforward methods for explaining their

predictions. For these models, (also known as black box models), approaches such as LIME and SHAP can be applied.

Model Interpretation with SHAP:

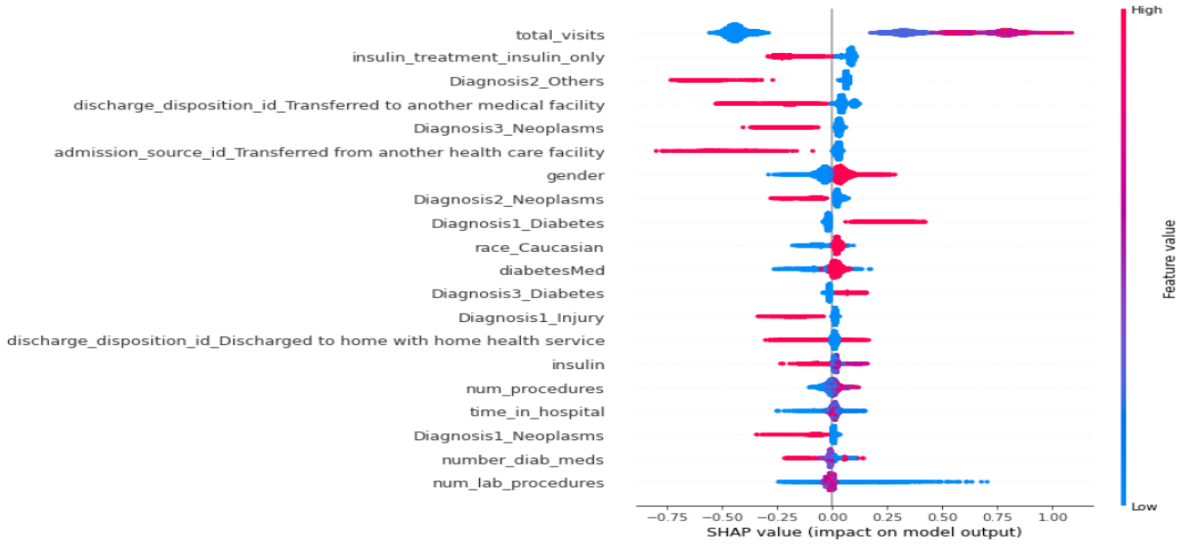
SHAP (Shapley Additive explanations) is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations, uniting several methods like ELI5, LIME etc. and representing the only possible, consistent and locally accurate additive feature attribution method based on expectations.

SHAP assigns each feature an importance value for a particular prediction. Its novel components include: the identification of a new class of additive feature importance measures, and theoretical results showing there is a unique solution in this class with a set of desirable properties.

Typically, SHAP values try to explain the output of a model (function) as a sum of the effects of each feature being introduced into a conditional expectation. Importantly, for non-linear functions the order in which features are introduced matters. The SHAP values result from averaging over all possible orderings. Proofs from game theory show this is the only possible consistent approach.

SHAP Summary Plot

Besides a typical feature importance bar chart, SHAP also enables us to use a density scatter plot of SHAP values for each feature to identify how much impact each feature has on the model output for individuals in the validation dataset. Features are sorted by the sum of the SHAP value magnitudes across all samples.



Findings: If the patient has the following characteristics, he has a high probability of being readmitted:

- High preceding year visits.
- If the patient is discharged to another medical facility or discharged to home with health services.
- High number of diagnoses.
- If the patient is given diabetes medicines.
- If the primary diagnosed disease was of circulatory system.

14. Conclusion:

Readmissions are acute, unplanned admissions to the hospital within a defined period of time from an initial admission. Readmission rates are a well-established health quality measure internationally as some readmissions that do occur are avoidable and if data is modelled correctly, groups of patients at high risk of readmission are identifiable. Hospital readmission is an important contributor to total medical expenditure and is an emerging indicator of quality of care. It is disruptive to patients and costly to healthcare systems. The objective of this project was to develop a predictive risk model to identify patients with diabetes who are at a high risk of hospital readmission.

This is done by analyzing key factors using machine learning methods and through retrospective analysis of patients' medical records which impact the all-purpose readmission of a patient with diabetes within 30 days and more than 30 days of discharge and comparing different classification models that predict readmission and evaluating the best model. In this project, the problem of predicting the risk of readmission was framed as a binary classification problem and several available prediction models were developed and evaluated.

This study has assessed how various data preprocessing techniques such as feature selection, missing value imputation and class balancing techniques may impact the results of prediction modeling using readmission for patients with a diabetes diagnosis as the context for the analysis. Then, various predictive models like Logistic Regression and Decision Tree were applied to this improved dataset (after pre-processing) to obtain risk of readmission predictions accuracy. The impact of different pre-processing choices was assessed on various performance metrics like Area under Curve (AUC), Precision, Recall and Accuracy. This study offers empirical evidence that most proposed models with selected pre-processing techniques significantly outperform the baseline methods with respect to selected evaluation criteria.

In this study, we evaluated various machine learning models to predict readmissions of high-

risk patients. Extending prior research, we performed class balancing considering the skewness of data. Our results show LightGBM slightly out-performing logistic regression and decision trees which were widely adopted in the literature. Some of the key features that drove readmissions are number of preceding year visits, length of stay, number of medications and number of diagnosis. Given the increasing cost of hospital readmissions and the increased emphasis on quality of care, the accuracy and validity of prediction models remain an important, yet elusive goal.

15. Business Recommendations

This project was designed to help hospitals decrease readmission rates for diabetic patients. With the proposed model, hospitals can target patients in high-risk percentiles. Not only are these patients at higher risk of being readmitted, the model precision is considerably better for the higher percentiles which means hospitals can efficiently use their resources to reduce readmission rates by administering medication that are highly effective in treating diabetes like insulin, metformin, glipizide and glyburide. Medical facilities can take precautionary measures with these patients during their initial admission by making A1C and maximum glucose serum test compulsory and providing the treatment accordingly. A follow-up visits to check their progress should also be schedule at the time of discharge.

This allows hospitals to provide a better quality of healthcare to their patients and also reduce the readmission rates. This reduction can help hospitals avoid penalties that are incurred for high readmission rates, leading to reduction in health expenditures for hundreds if not thousands of dollars per diabetes patient, while simultaneously improving health outcomes, and saving lives.

To summarize, these are the recommendations proposed to the business client:

- Medical facilities can take precautionary measures with patients during their initial admission by making A1C and maximum glucose serum test compulsory and providing the treatment accordingly reason being 80-90% of the readmitted patients had not gone under these tests.
- A follow-up with the discharged patients should be one to keep a track of their health and to counsel them time-to-time.
- High-risk patients' current medicines' regime should be re-evaluated and the most effective medicines should be considered.
- Most Effective Medicines as per the findings are Insulin, Metformin, Glipizide. These medicines are coming out to be statistically significant, coming out to be quite important with respect to different machine learning models employed, are most

widely prescribed and are associated with low risk of readmission if given to the patient.

- The annual plans, financials and infrastructure / inventory of the hospital should be planned accordingly by taking into account the predicted readmissions.
- Hospitals must provide extra attention and care to high-risk patients.

16. Future Scope:

This research study has only targeted patients with diabetes. Readmission prediction model needs to be generated for other key health conditions also such as Heart disease, kidney disease etc. in Indian Healthcare system. In the future studies, planned and unplanned (emergency) readmissions needs to be considered.

Various other key features in the medical records, like family history (to find hereditary information), emotional status (depression), socioeconomic status, and lifestyle habits (exercise), smoking status and season of readmission need to be collected and analyzed. It will be interesting to perform a more exhaustive exploration of additional features in the dataset and study their relevance towards predicting the risk of readmission.

17. References and Links

- Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records," BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.
  Website: https://www.hindawi.com/journals/bmri/2014/781670/

- The Hospital Readmissions Reduction (HRR) Program, Center for Medicare and Medicaid Services.
  Website:https://www.cms.gov/Medicare/Quality-Initiatives-Patient-Assessment- Instruments/Value-Based-Programs/HRRP/Hospital-Readmission-Reduction- Program

- Diabetes 130-US hospitals for years 1999-2008 Data Set provided by Centre for Clinical and Translational Research, Virginia Commonwealth University.

  Website:https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008