A

Capstone Project Report

On

"PREDICTING RE-ADMISSION OF PATIENTS' WITH DIABETES"

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

PGP in Data Science and Engineering

Affiliated to

Great Lakes Institute of Executive Learning

(Session 2020)

Guided by-

Mr. Animesh Tiwari

Submitted by-

Aniket Nikhar

Anurag Wasnik

Himanshu Jaisal

Khushboo Thapa

Supriya Hadawale

Acknowledgment

I would like to express my special thanks of gratitude to my Great Learning teachers as well as my mentor Mr. Animesh Tiwari who gave me the golden opportunity to do this wonderful project on the topic ***Predicting Diabetic Patient Readmission***, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

Secondly I would also like to thank my project partners who helped me a lot in finalizing this project within the limited time frame.

Project Summary

Batch details	Mumbai July 2020
Team members	Aniket Nikhar Anurag Wasnik Himanshu Harilal Jaisal Khushboo Puran Thapa Supriya Sameer Hadawale
Domain of Project	Healthcare
Proposed project title	Predicting Re-admission of Patients' with Diabetes
Group Number	2
Team Leader	Khushboo Puran Thapa
Mentor Name	Animesh Tiwari

Date: 22/04/2021

Signature of the Mentor

Signature of the Team Leader

Table of Contents

Project Details

OVERVIEW

Business problem statement (GOALS)

1. Business Problem Understanding

Diabetes Mellitus (**DM**), commonly known as **diabetes**, is a group of metabolic disorders characterized by a high blood sugar level over a prolonged period of time. Diabetes is a number of diseases that involve problems with the hormone insulin. Normally, the pancreas (an organ behind the stomach) releases insulin to help your body store and use the sugar and fat from the food you eat. Diabetes occurs when one of the following occurs:

- When the pancreas does not produce any insulin
- When the pancreas produces very little insulin
- When the body does not respond appropriately to insulin, a condition called "insulin resistance"

Diabetes is a lifelong disease. Approximately 18.2 million Americans have the disease and almost one third (or approximately 5.2 million) are unaware that they have it. An additional 41 million people have pre-diabetes. As yet, there is no cure. People with diabetes need to manage their disease to stay healthy. There are three main types of diabetes mellitus:

- Type 1 diabetes results from failure of the pancreas to produce enough insulin due to loss of beta cells. This form was previously referred to as "insulin-dependent diabetes mellitus" (IDDM) or "juvenile diabetes". The loss of beta cells is caused by an autoimmune response. The cause of this autoimmune response is unknown.
- Type 2 diabetes begins with insulin resistance, a condition in which cells fail to respond to insulin properly. As the disease progresses, a lack of insulin may also develop. This form was previously referred to as "non-insulin-dependent diabetes mellitus" (NIDDM) or "adult-onset diabetes". The most common cause is a combination of excessive body weight and insufficient exercise.
- Gestational diabetes is the third main form, and occurs when pregnant women without a previous history of diabetes develop high blood sugar levels.

As of 2019, an estimated 463 million people had diabetes worldwide (8.8% of the adult population), with type 2 diabetes making up about 90% of the cases. Rates are similar in women and men. Trends suggest that rates will continue to rise. Diabetes at least doubles a person's risk of early death. In 2019, diabetes resulted in approximately 4.2 million deaths. It is the 7th leading cause of death globally. The global economic cost of diabetes related health expenditure in 2017 was estimated at US$727 billion. In the United States, diabetes cost nearly US$327 billion in 2017. Average medical expenditures among people with diabetes are about 2.3 times higher.

2. Business Objective

Hospital readmission is a high-priority health care quality measure and target for cost reduction, particularly within 30 days of discharge (30-day readmission, aka early readmission). Despite the broad interest in readmission, relatively little research has focused specifically on readmission of patients with diabetes. The burden of diabetes among hospitalized patients, however, is substantial, growing, and costly, and readmissions contribute a significant portion of this burden. Reducing readmission rates among patients with diabetes has the potential to greatly reduce health care costs while simultaneously improving care. Recent research has provided some insight into the risk factors for readmission and the barriers to reducing readmission risk, as well as ways to mitigate that risk.

Many believe readmission rates reflect the quality of health care delivery, although this is debated. In order for readmissions to be prevented, they must first be understood. A good understanding of the causes and risk factors for readmission is crucial. Objective is readmission risk reduction based on concrete understanding after identifying the factors that lead to the high readmission rate of diabetic patients post discharge or not so that the quality of care can be improved along with improved patient's experience, health of the population and reduce costs by lowering readmission rates. Also, to identify the medicines that are the most effective in treating diabetes.

3. Approach

- Roadmap

Understanding the business problem recorded in the dataset and the multiple variables that are implemented to explain the challenge. Examining the meaning behind the variables, in our case, the medical terminologies and nomenclatures used. To better understand these terms, we delved deep into research papers on diabetes and the complications that follow along.

- Data Extraction

Dataset was obtained and inspected in every combination possible to extract the insights recorded within.

- Data Cleaning

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Data cleansing maybe performed interactively with data wrangling tools or as batch processing through scripting. After cleansing, a data set should be consistent with other similar data sets in the system.

- <u>Exploratory Data Analysis</u>

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions. EDA is primarily used to see what data can reveal beyond the formal modelling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

- <u>Preprocessing/Feature Engineering</u>

Data preprocessing is an umbrella term that covers an array of operations data scientists will use to get their data into a form more appropriate for what they want to do with it. In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or encoded to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

- <u>Statistical Findings</u>

The core of machine learning is centered around statistics. You can't solve real-world problems with machine learning if you don't have a good grip of statistical fundamentals. From exploratory data analysis to designing hypothesis testing experiments, statistics play an integral role in solving problems across all major industries and domains.

Statistics is a set of mathematical methods and tools that enable us to answer important questions about data. It is divided into two categories:

1. **Descriptive Statistics** - this offers methods to summarize data by transforming raw observations into meaningful information that is easy to interpret and share.
2. **Inferential Statistics** - this offers methods to study experiments done on small samples of data and chalk out the inferences to the entire population (entire domain).

- <u>Feature Selection</u>

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons:

1. Simplification of models to make them easier to interpret by researchers/users
2. Shorter training times
3. To avoid the curse of dimensionality
4. Enhanced generalization by reducing overfitting (formally, reduction of variance)

- Evaluate and deploy Model

This makes for one of the most important steps as the machine learning algorithm helps build a workable data model. There are many algorithms to choose from. Once the right machine learning algorithm is settled on, next comes its evaluation. Techniques such as cross-validation or even ROC (Receiver operating characteristic) curve, work well for generalizing the model output for new data. If the model appears to be producing satisfying results, model can be implemented and make a difference to the business problem.

4. Conclusions

Hospital readmission of patients with diabetes is an important health care quality measure and driver of costs. Major risk factors for readmission include lower socioeconomic status, racial/ethnic minority, greater burden of comorbidities, public insurance, emergent or urgent admission, and a history of recent prior hospitalization. Certain hospitalized patients with diabetes may be at higher risk of readmission than those without diabetes. Multiple health system and patient-related barriers to reducing readmission rates exist. A mix of expert opinion and a handful of mostly small studies provide a number of potential strategies for reducing readmission risk, including inpatient education, specialty care, better discharge instructions, coordination of care, and post-discharge support. The diabetes-specific strategies such as diabetes education, intensifying therapy, and outpatient diabetes care tend to be more effective in poorly controlled patients and tend to reduce.

TOPIC SURVEY IN BRIEF

1. **Problem understanding**

To identify the factors that lead to the high readmission rate of diabetic patients within 30 days post discharge and correspondingly to predict the high-risk diabetic-patients who are most likely to get readmitted within 30 days so that the quality of care can be improved along with improved patient's experience, health of the population and reduce costs by lowering readmission rates. Also, to identify the medicines that are the most effective in treating diabetes.

2. Current solution to the problem

The current medical environment, being organized and assessed by coded input into electronic medical records systems, limits comprehensive evaluation of the many factors of patient readmission, resulting in potentially misleading and poorly representative metrics.

Given the potential consequences of the Hospital Readmissions Reduction Program on hospital systems and the near epidemic levels of DFIs, one should consider ways to reduce risks to the patient and the health care system overall.

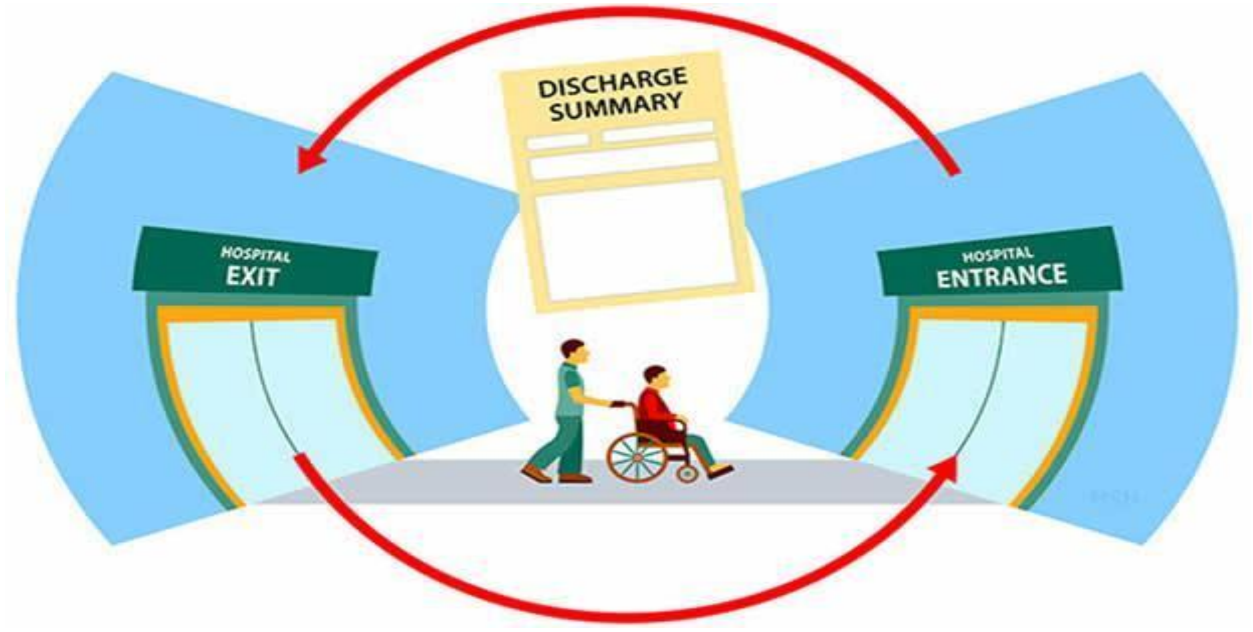
Inpatient classification systems. Upon the initial evaluation during a patient hospitalization, the utilization of a validated classification system (such as the Society for Vascular Surgery Wound, Ischemia and Foot Infection (WIFi) system) in relaying and stratifying clinical findings can relay clinical correlation with prognosis and initial hospitalization length of stay to those involved in patient care. Additionally, because the WIFi classification also aids in the assessment of healing potential, a higher (worse) score on this system may be an early indicator to the care team as to the risk of future rehospitalization. The utilization of the WIFi classification system, in particular, promotes repetitive and reproducible evaluations of critical parameters of a DFI (extent of the wound, perfusion, extent of infection) by new and experienced practitioners alike, theoretically reducing some of the risk involved with inadequate physical examination, effectively reducing the risk of patient return for further workup or treatment.

Clear and concise communications. Patients with longstanding diabetes demonstrate increased susceptibility to early cognitive decline, impaired verbal understanding and multiple forms of diminished mentation relative to their non-diabetic counterparts. Upon discharge, providing basic yet helpful information, including a written warning as to the signs and symptoms of infection, readily visible contact information, plain instructions for dressing changes and defining the next office visit and emergency bandage instructions are simple ways to decrease the loss of information between discharge and an outpatient evaluation. Whenever possible, the discussion of patient care with family members alongside patients has demonstrated improved patient adherence and outcome.21Finally, it may be necessary to lower one's threshold for ordering home health care on discharge in an effort to promote adherence and reduce readmission.

The checks and balances of a team approach. Modern inpatient care of the DFI requires the collaboration of a comprehensive diabetic foot team including, at a minimum, podiatrists, vascular surgeons and infectious disease specialists. This combined approach allows specialists to independently evaluate and participate in the treatment of the critical parameters of patient care. The redundancy in physical examination inherent in a team approach reduces oversight and improves outcomes in the care of the DFI, which may correlate with decreased readmission.

3. Proposed solution to the problem

After feature engineering and isolating significant variables, try to fit multiple models to test the classification metrics. The solution to this problem will be described in detail, starting with EDA and then feature engineering and the models fit on the variables.



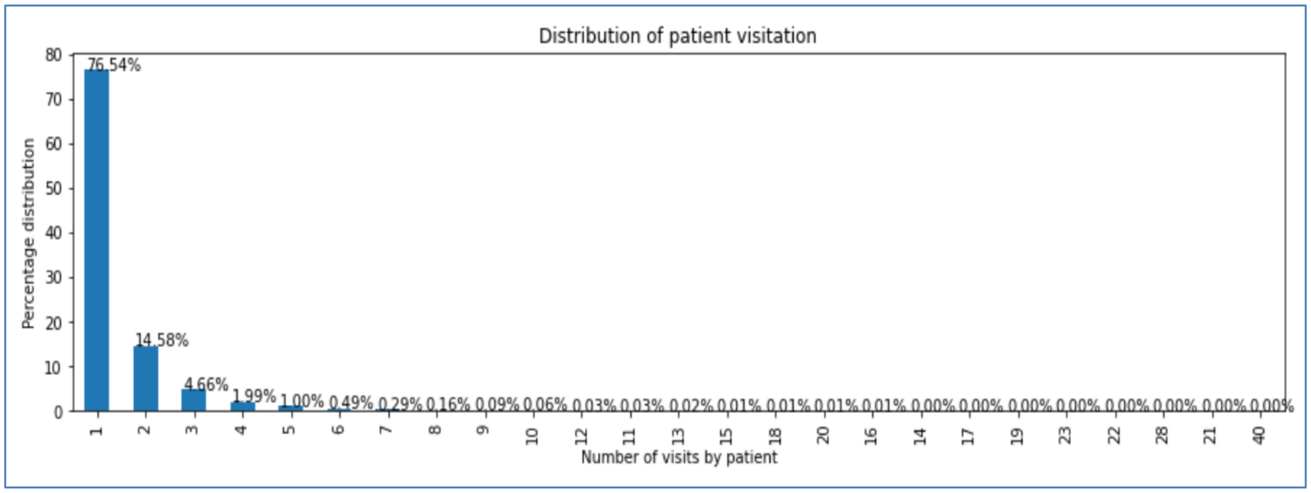
Data Set Description:

Encounter ID	Unique identifier of an encounter
Patient number	Unique identifier of a patient
Race Values	Caucasian, Asian, African American, Hispanic, and other
Gender	male, female, and unknown/invalid
Age	Grouped in 10-year intervals
Weight	Weight in pounds
Admission type	9 distinct values(Depends on patient situation)
Discharge disposition	29 distinct values(discharged to home, expired, and not available
Admission source	21 distinct values(Depends on patient)
Time in hospital	Days between admission and discharge
Payer code	23 distinct values(Payment code)
Medical specialty	Specialty of the admitting physician

Number of lab procedures	Lab tests performed
Number of procedures	Procedures performed during the encounter
Number of medications	Generic names administered during the encounter
Number of outpatient visits	Outpatient visits within the year
Number of emergency visits	Emergency visits within the year
Number of inpatient visits	Inpatient visits within the year
Diagnosis 1	Primary diagnosis
Diagnosis 2	Secondary diagnosis
Diagnosis 3	Additional secondary diagnosis
Number of diagnoses	No of diag. entered to the system 0%
Glucose serum test result	Prediabetic test(Range: >200, >300,normal,none)
A1c test result	Test for diabetic(Range: >8%, >7%, >7%-<8%)
Change of medications	Change in diabetic medications
Diabetes medications	Diabetic medication prescribed
24 features for medications	24 different drugs that are used in treatment
Readmitted	Days to inpatient readmission

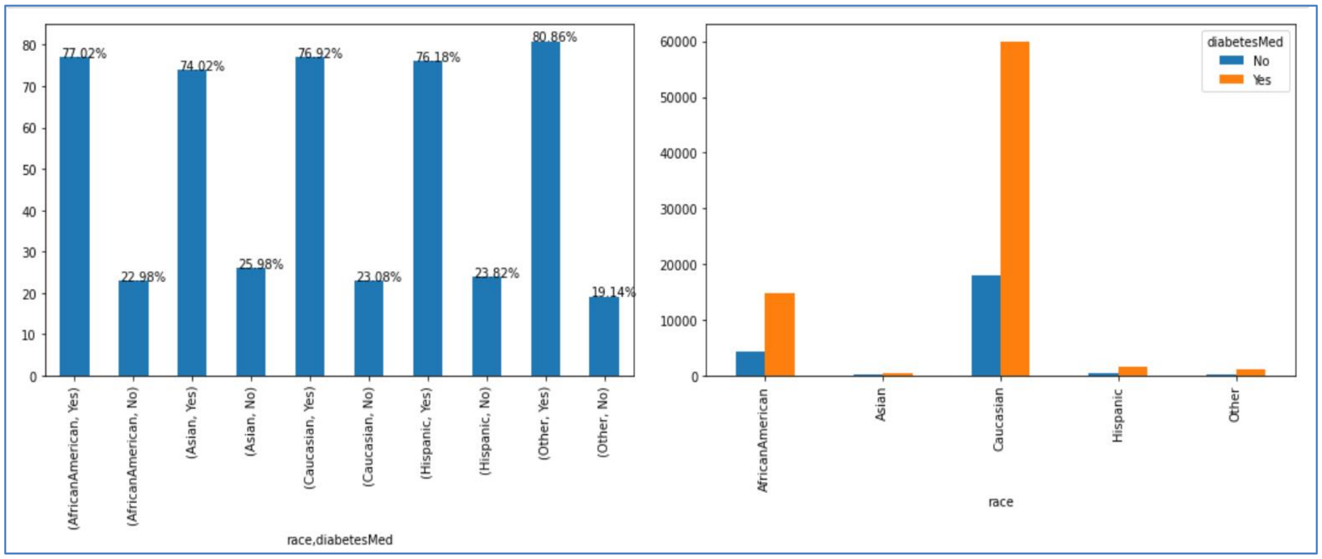
Exploratory Data Analysis:

Patient visitation:



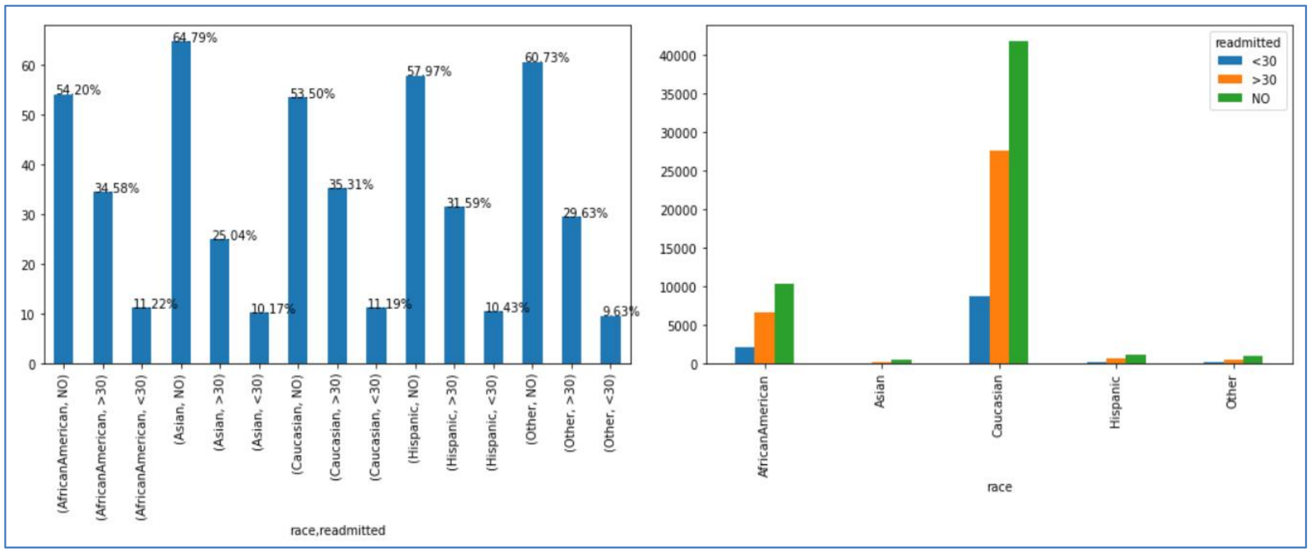
- Out of 71,456 Patients, 16,758 (23.45%) Patients Revisited the Hospital
- 76.54% Patients Visited Single Time in the Hospital
- 23.45% Patients Revisited the Hospital.
- The highest number of revisits made by patient: 40 (though this was unique case)

Race vs Diabetes Med:



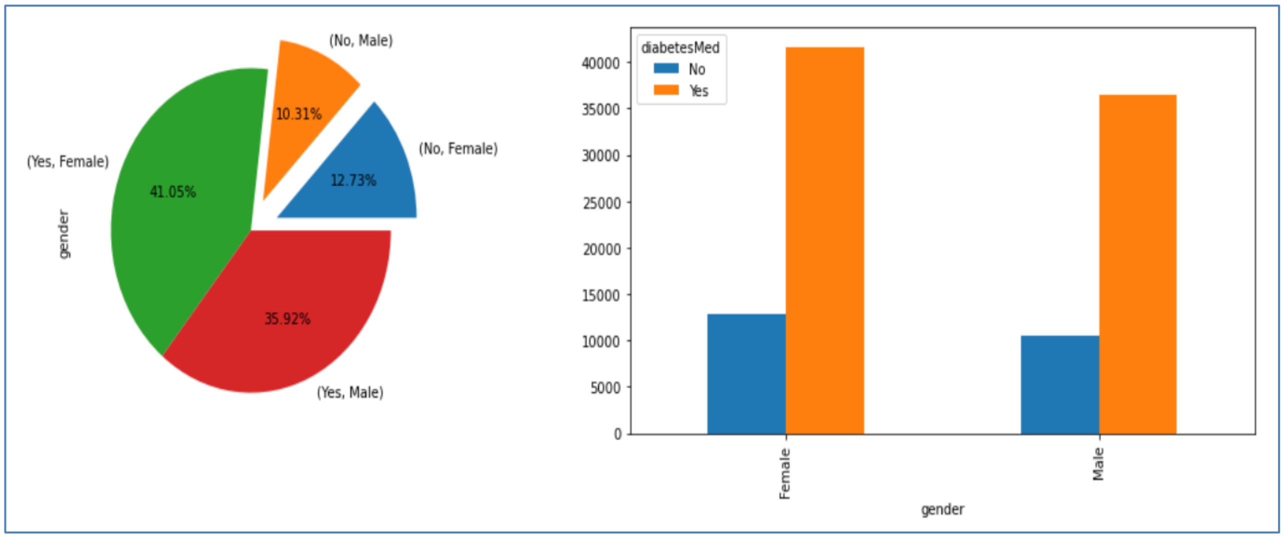
- Majority people visiting the hospital are having Diabetes Meds
- There is a consistency between 75-80% Patients are consuming Diabetic Meds irrespective of their Race.

Race VS Readmission:



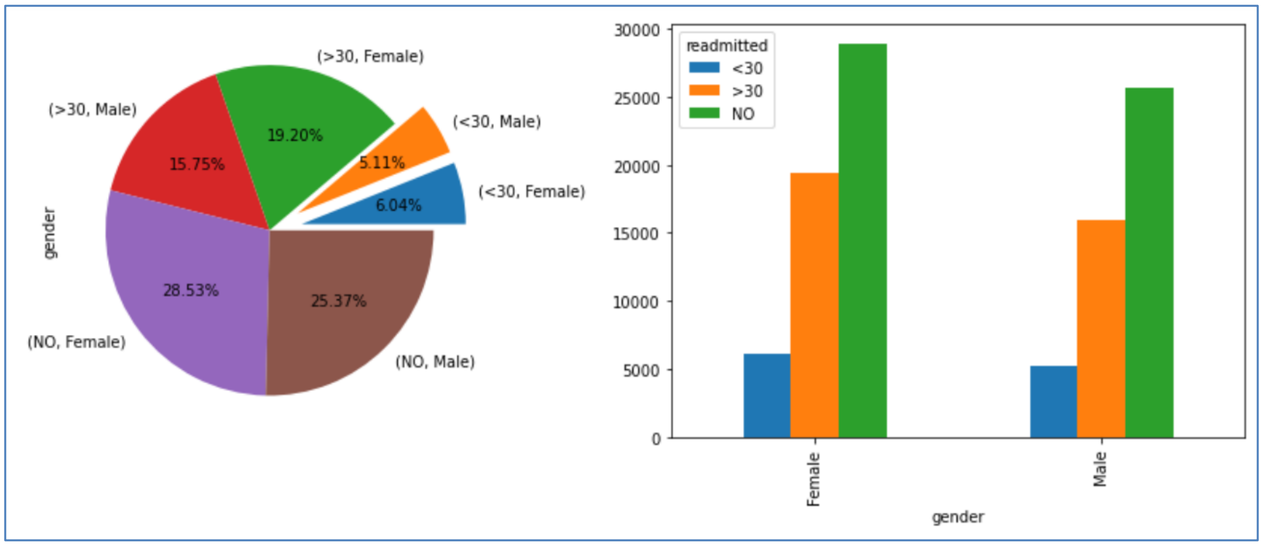
- Highest proportion of patients are Caucasians, followed by African Americans.
- There is 10% chance for patient being readmitted within 30 days and 30% chance of being readmitted after 30 days irrespective of their ethnicity.
- There is 60% chance across all the race that patient was 'Not Readmitted' again.

Gender:



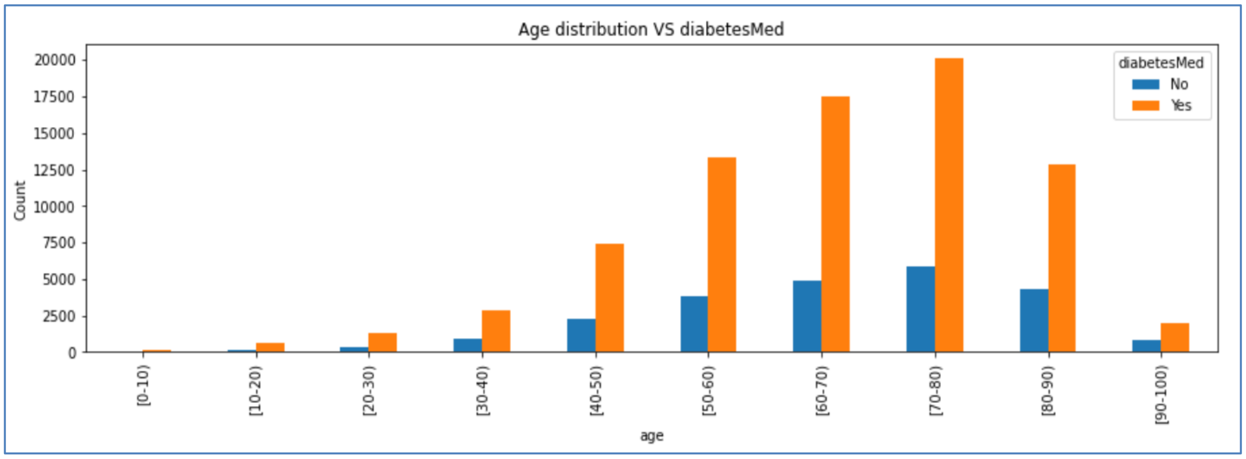
- There is a higher proportion of female patients than male, 54516 females and 46868 males.
- Most patients visiting hospital are prescribed Diabetes Meds irrespective of Gender.
- Out of Total Patients, around 23% Patients were not given Diabetes Meds.
- Both genders have 23% not consuming Diabetes Meds.

Gender vs Readmission:



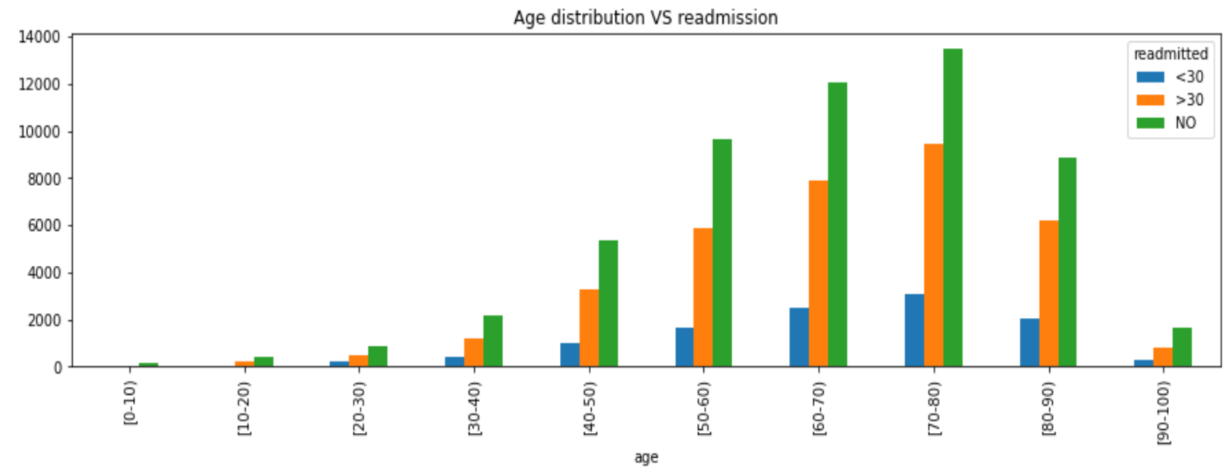
- More female patients were readmitted after 30 days than male patients.
- Similar trend is seen in readmission within 30 days in the pie chart.

Age vs Diabetes Meds:



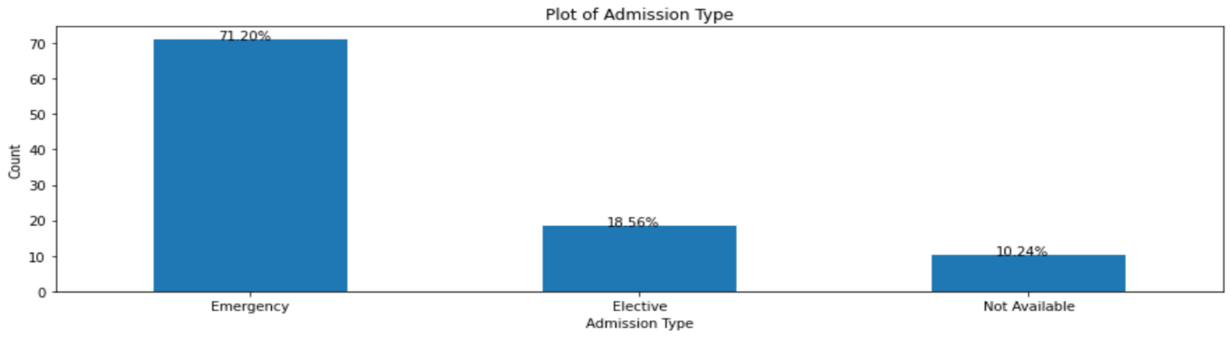
- Around 75% of patients across all age groups are consuming Diabetes Meds.

Age vs Readmission:



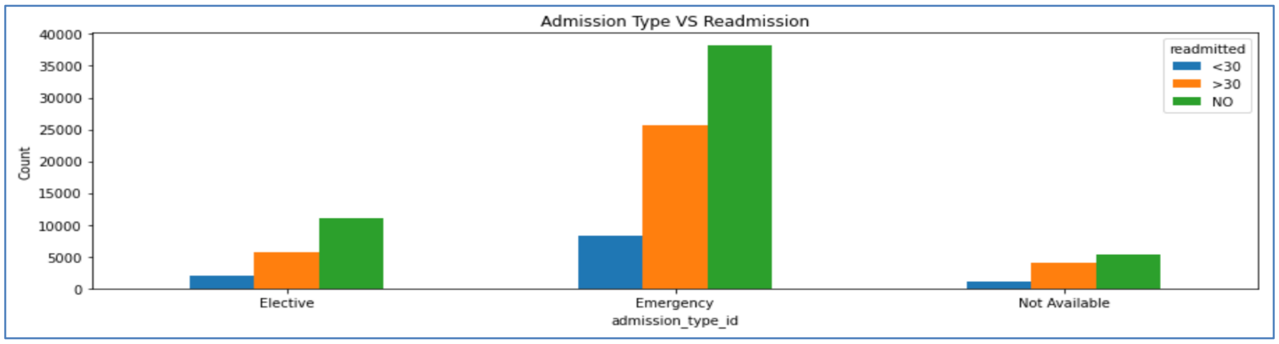
- Age group of 70-80 are more likely to be readmitted and age group of 60-90 are more prone to being hospitalized. Overall Readmission rate is low

Admission Type:



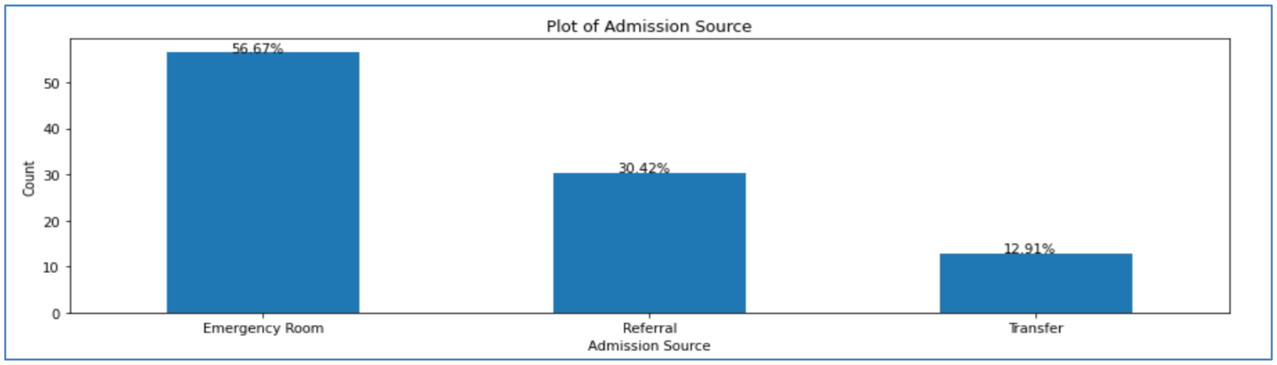
- Emergency Room alone amounts to 71% of total Admission Type
- Elective contributes to 20% of total Admission Type and 10% data is ‘Not Available’

Admission Type vs Readmission:

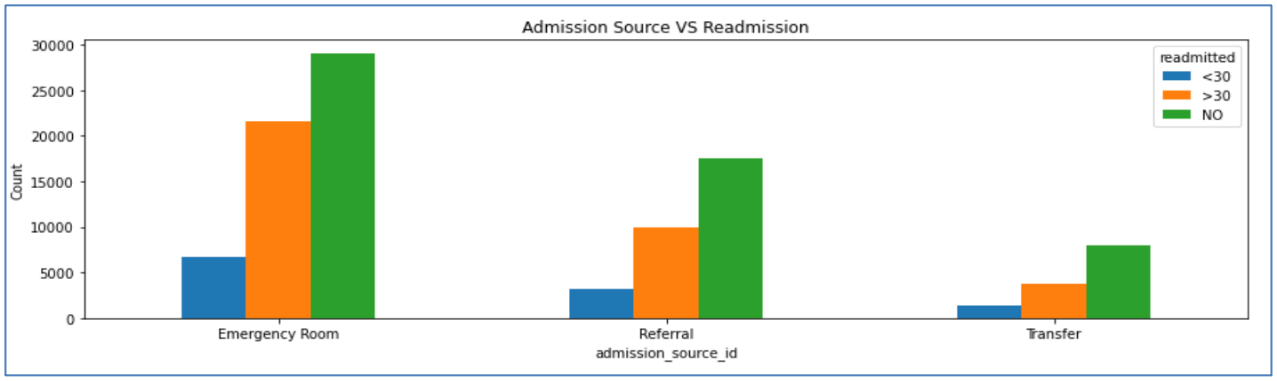


● Emergency has the highest count of patients not being readmitted, but also being readmitted. We do have a significant part which is unavailable.

Admission Source ID:

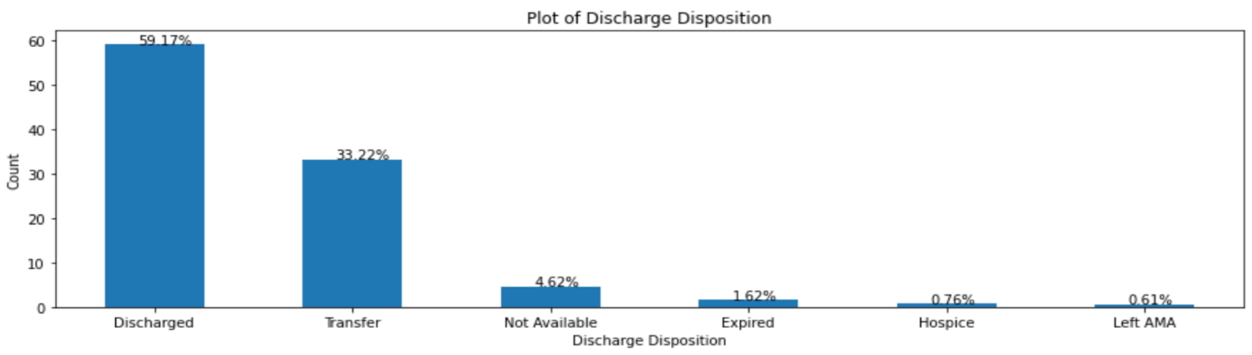


● Emergency has 56% of total Admission Source, followed by Referral 30%.
● Transfer has the lowest count of 12%.



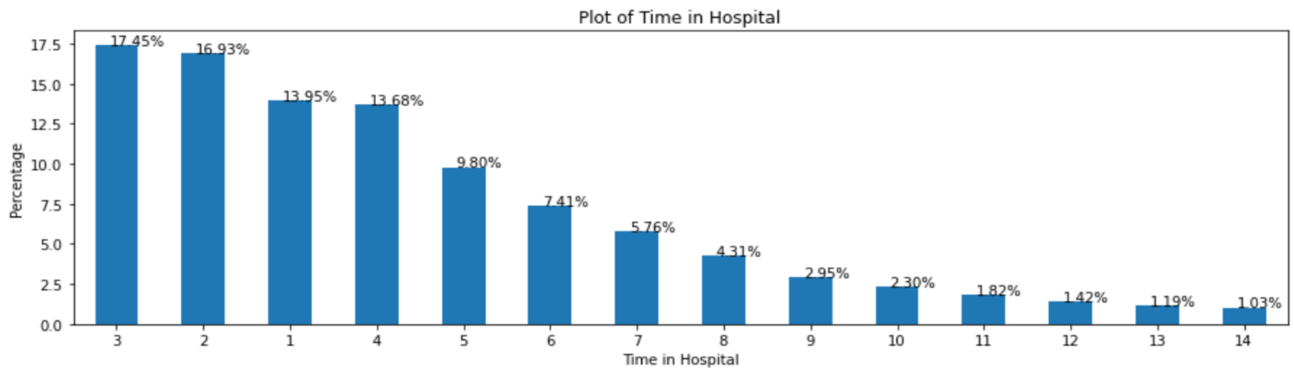
● Patients Readmitted in the Hospital has a Highest Frequency counts from Emergency Room, Followed by Refferal

Discharge Disposition:



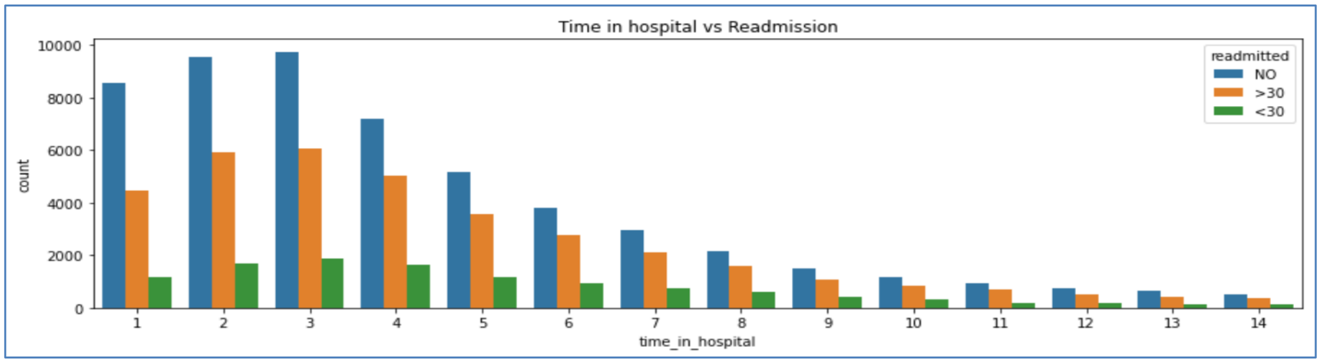
- Discharged amounts to 60%, followed by 'Transfer' 30%.
- Rest is either Left AMA or Hospice

Time in Hospital:



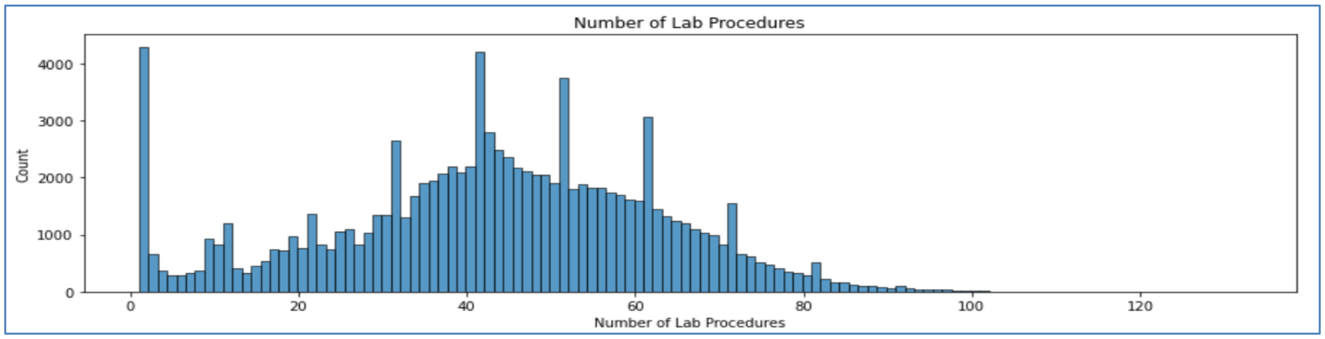
- Average time in hospital is 3 to 6 days.
- 17.5% patients spent 3 days.
- 17.0% patients spent 2 days.
- 14.0% patients spent 1 day in hospital.

Time in hospital Vs Readmission:



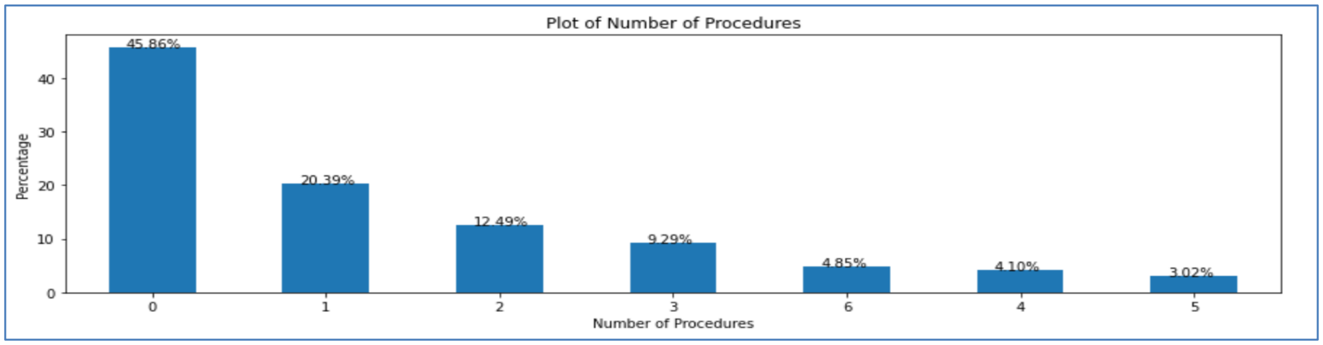
- Most likely time spent by patient is 2-3 days without readmission.
- However, patients readmitted also spent 2-3 days in hospital.

Number of Lab Procedures:



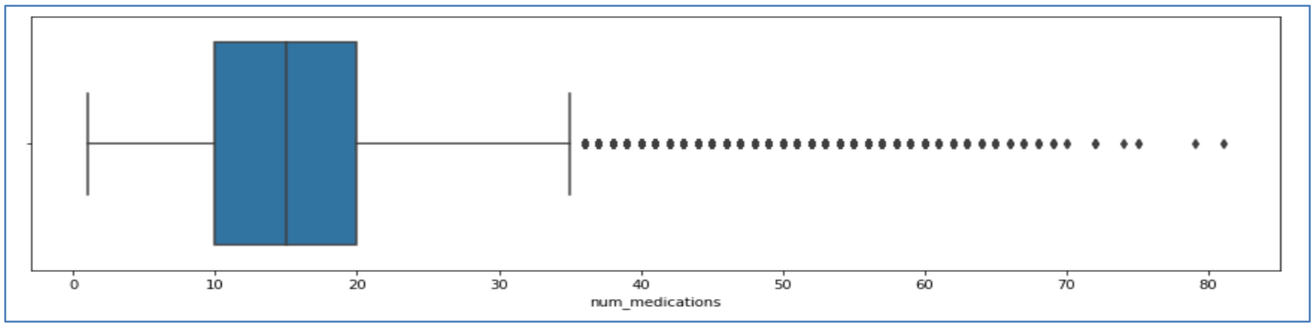
- Average of 35-55 Lab Procedures are conducted.
- Despite the variation in the graph, the highest number of Lab procedures is still 1.

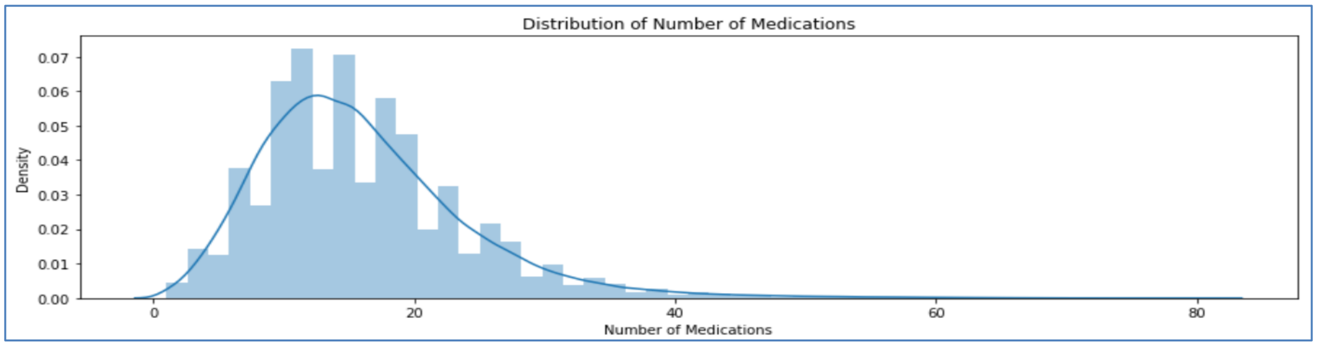
No of Procedures:



- 45% of times ‘No’ additional procedures were conducted.
- 20% of times ‘1’ and 13% of times ‘2’ additional procedures were conducted.

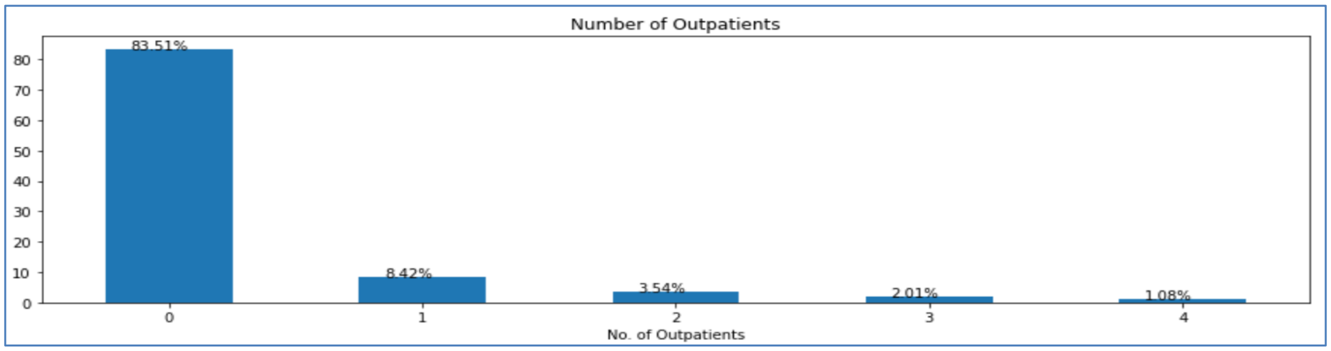
Number of Medications:

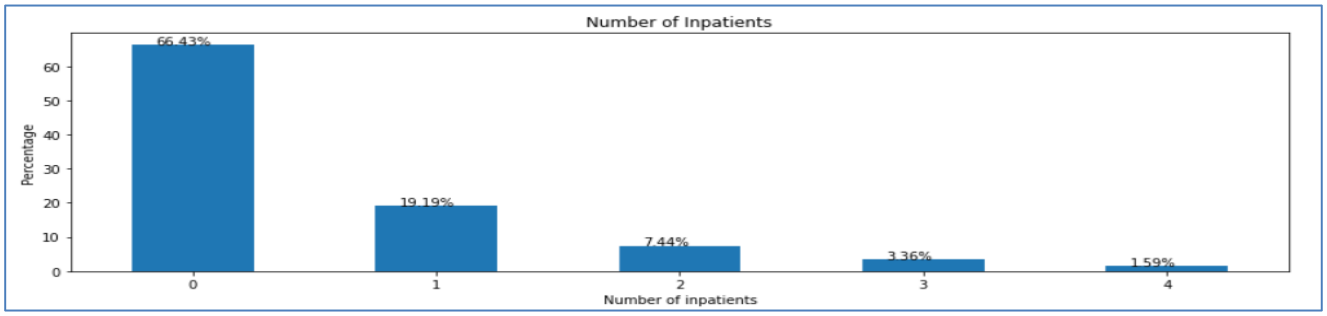




- An average of 10-20 medications are prescribed toward treatments.
- The number of medications vary from 1 – 40, however some patients were given 80 medicines.

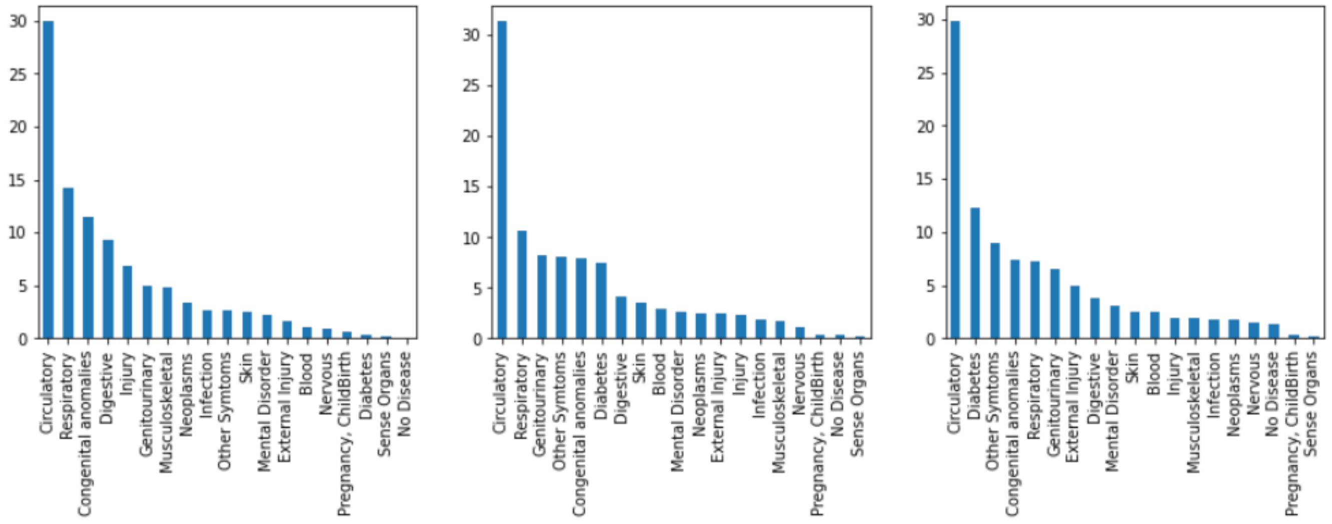
Number of Outpatients and Inpatients:





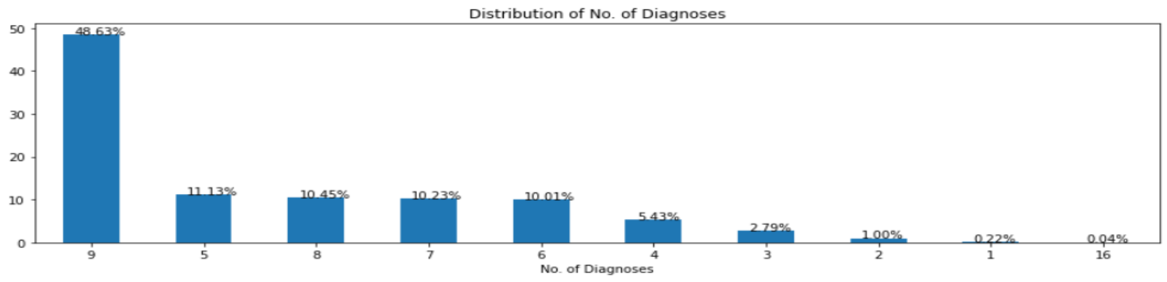
- In the case of Outpatient, there is 83% chance of outpatient addressal in 1 appointment with the doctor and 17% chance that the patient needs to revisit OPD again.
- With Inpatient category, there is 63% chance of that patient ailment relief is achieved with 1 appointment with the doctor and 37% chance of needing a revisit.

<u>Diagnoses:</u>



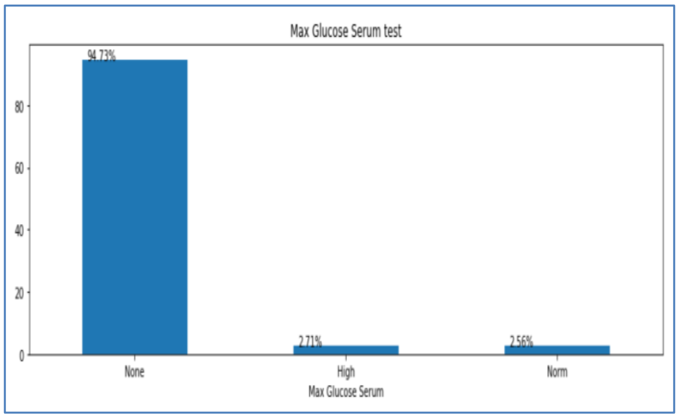
- **Diagnosis 1** The primary diagnosis (coded as first three digits of ICD9); 848 distinct values
- **Diagnosis 2** Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
- **Diagnosis 3** Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values

- With Increasing Diagnosis Advancements, We can observe Increasing Trends in the Diabetes
- Primary Diagnosis → Diabetes is at 17th Position
- Secondary Diagnosis → Diabetes is at 6th Position
- Additional Secondary Diagnosis → Diabetes is at 2nd Position

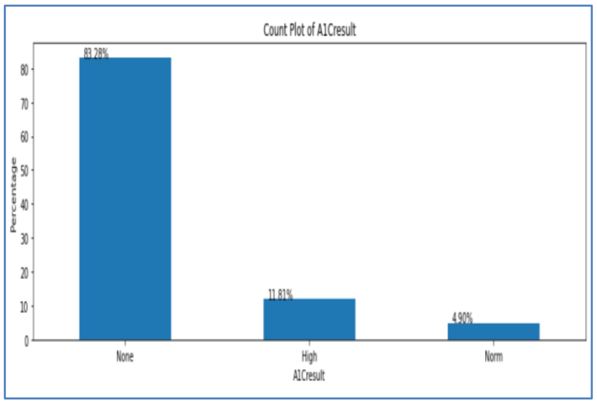


1. 48.63% patients with 9 diagnoses
2. 11.13% patients with 5 diagnoses
3. 10.45% patients with 8 diagnoses
4. 10.23% patients with 7 diagnoses
5. 10.01% patients with 6 diagnoses
6. 5.43% patients with 4 diagnoses
7. 2.79% patients with 3 diagnoses
8. 1.00% patients with 2 diagnoses
9. 0.22% patients with 1 diagnosis
10. 0.04% patients with 16 diagnoses

<u>Max Glucose serum test and A1C result Test:</u>



Max Glu Serum



A1c Result

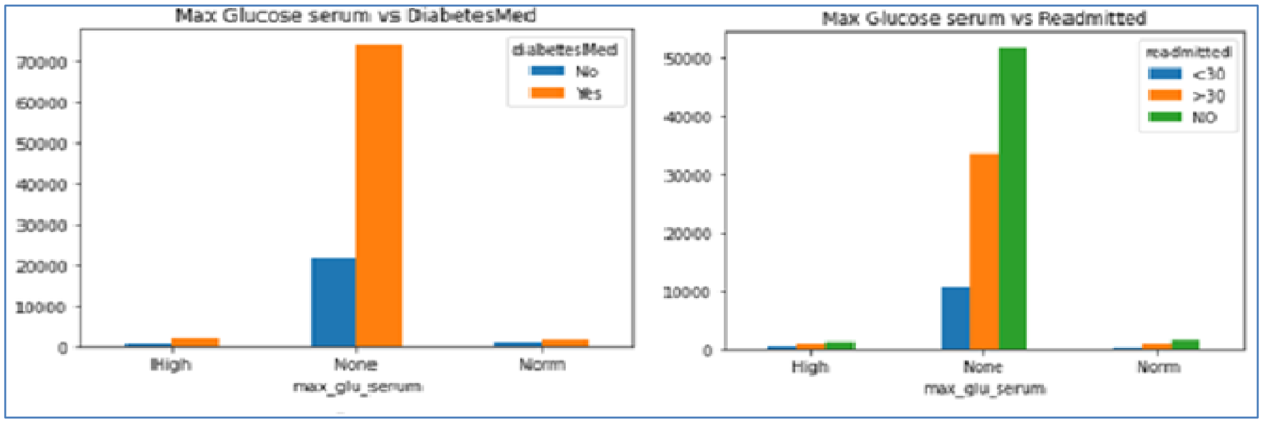
Max Glucose Serum test:

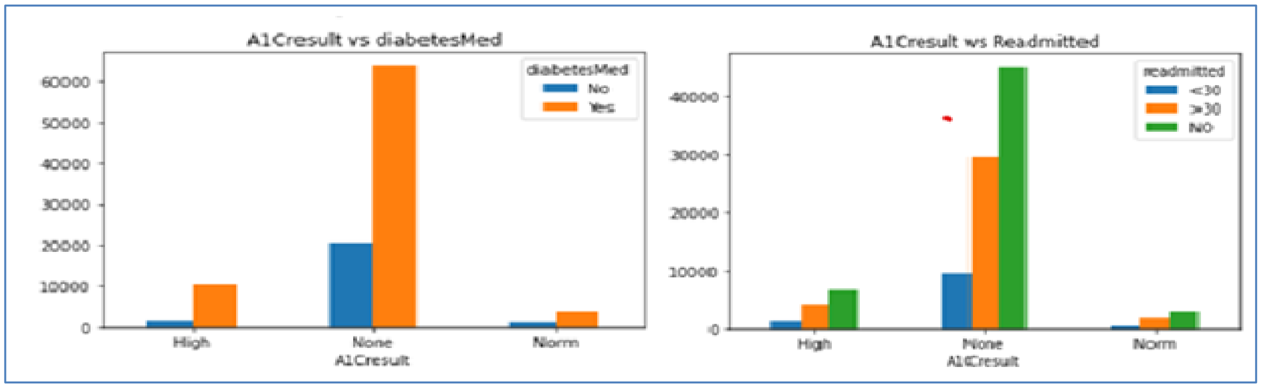
- Highest subcategory is 'None' - 94.75 % or 96329 records - indicating test was not conducted, followed by 'High' and closely followed by 'Norm', which indicates results of the test as 'High' and 'Normal' levels respectively.

A1C result:

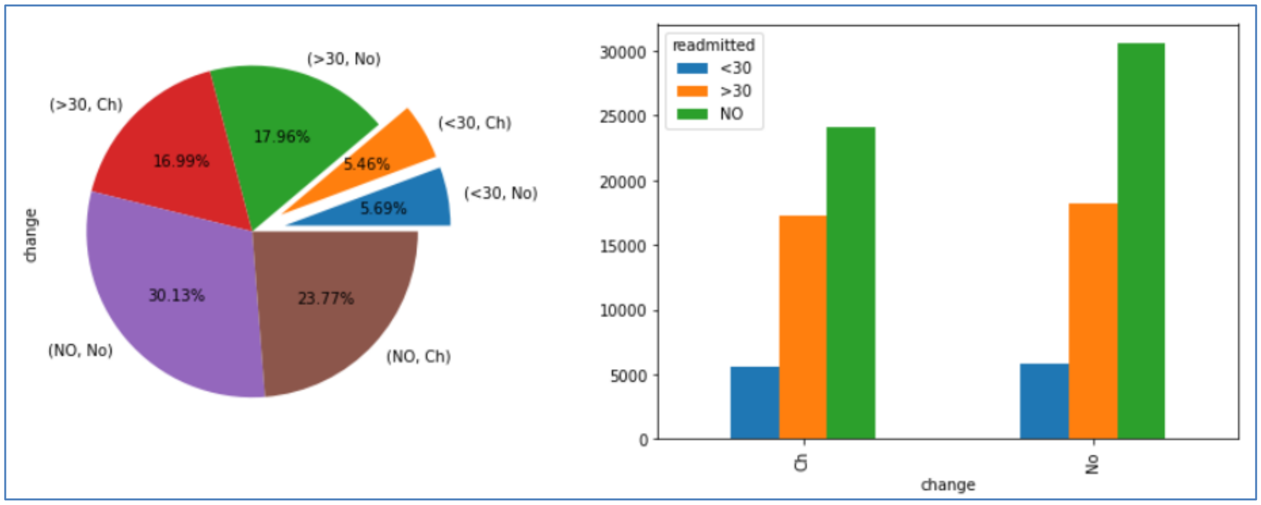
- Highest subcategory is 'None' - indicating test was not conducted, followed by 'High' and followed by 'Norm', which indicates results of the test as 'High' and 'Normal' levels respectively.

Max Glucose serum/A1C Result vs Diabetes Med and Readmission:



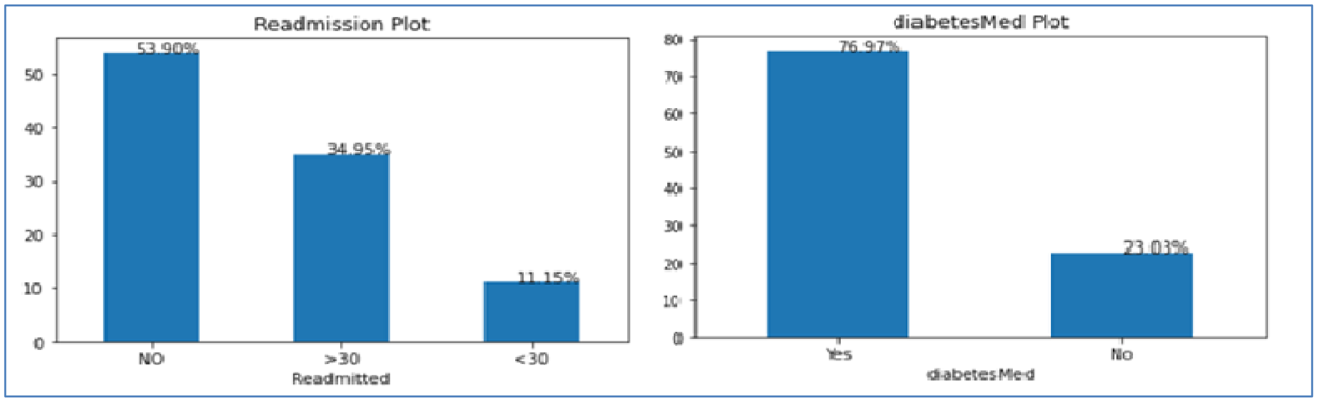


Change in Medication Vs Readmission:



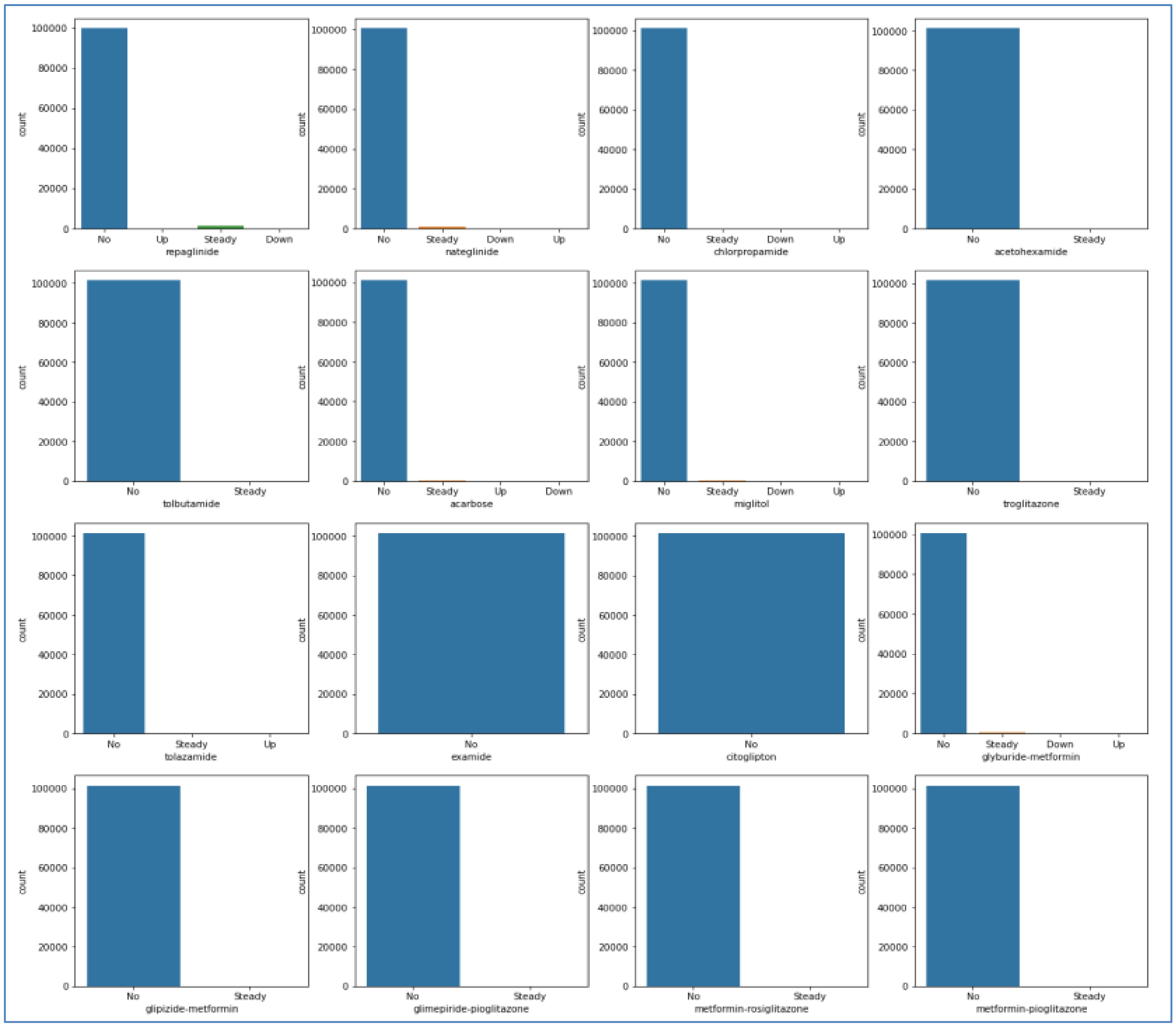
- Change in medications are not observed for Non-Diabetic patients
- There is not a single instance where patients were prescribed more medication and asked to continue with the medication they had earlier.

Readmission and Diabetes plot:



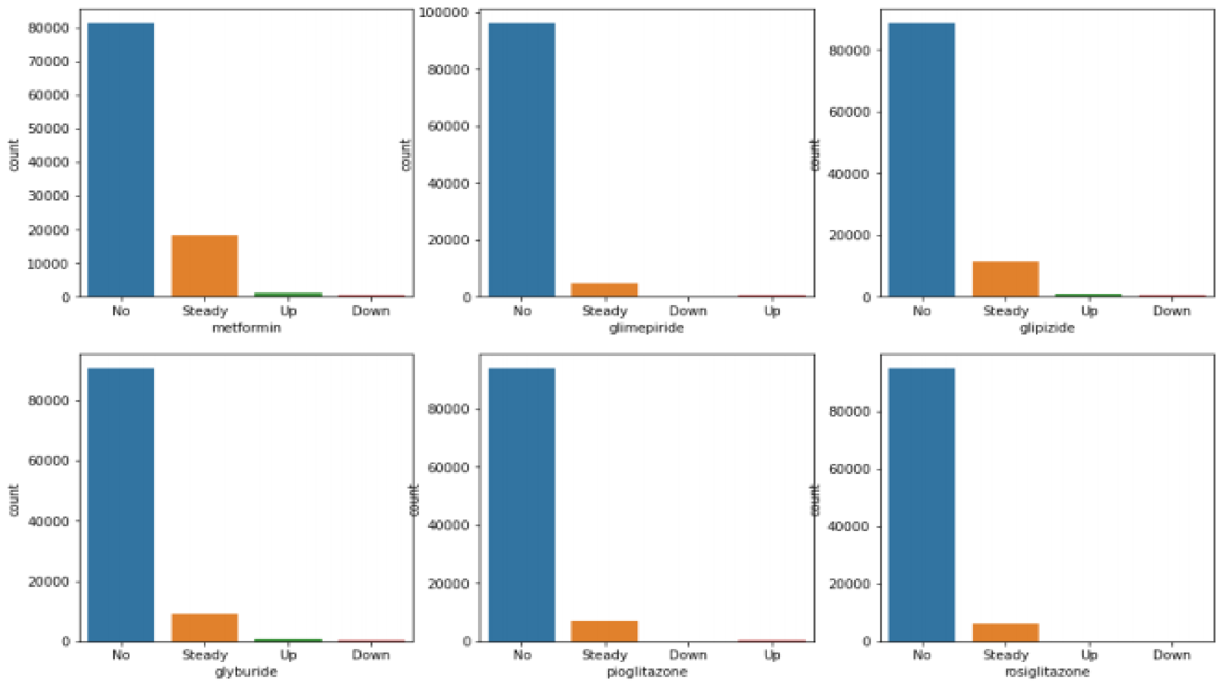
- 55% Patients were not Readmitted, 45% were readmitted, 35% Patients have been readmitted more than 30 times and finally 10% have been readmitted Less than 30 times.
- Almost 77% were prescribed diabetes medication and a mere 23% not prescribed diabetes medication.

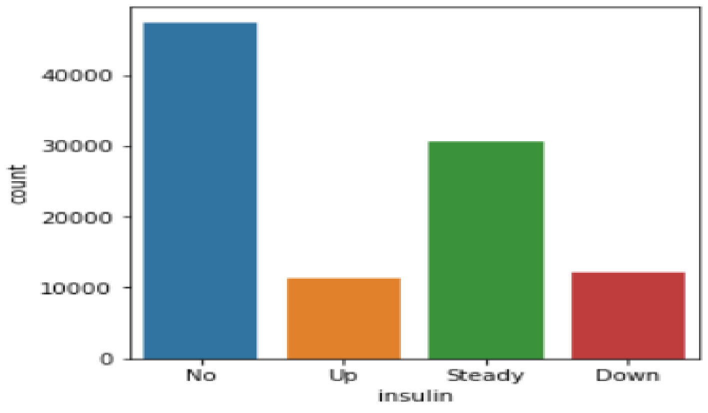
Medicines:



These columns (all variables are medications prescribed for managing diabetes) have high im-balance, in many cases one category highly outnumbering the other.

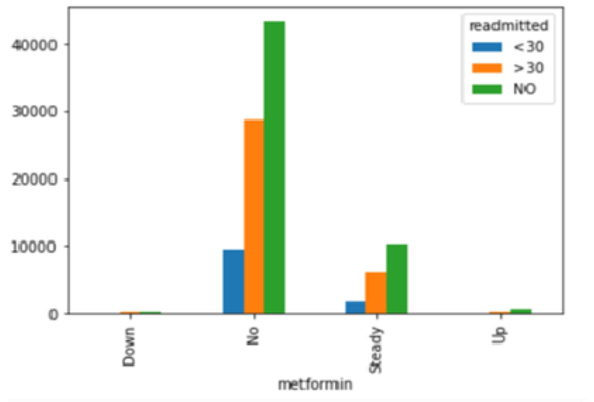
There are "7" Features having contribution of less than 98% for single class in a feature

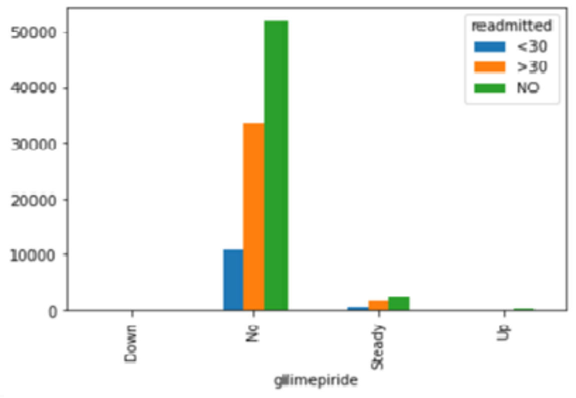


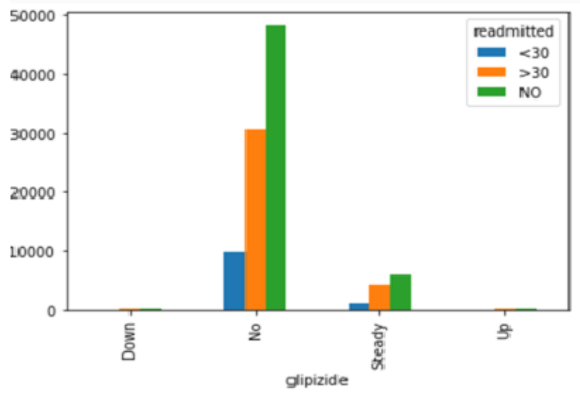


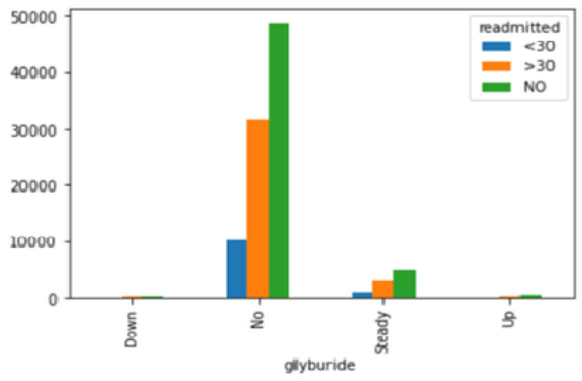
- Although the majority readings are not so high for Class 2, but it is also not so less to be neglected.
- We would keep the option of treating it at later stages to keep an eye on Accuracy

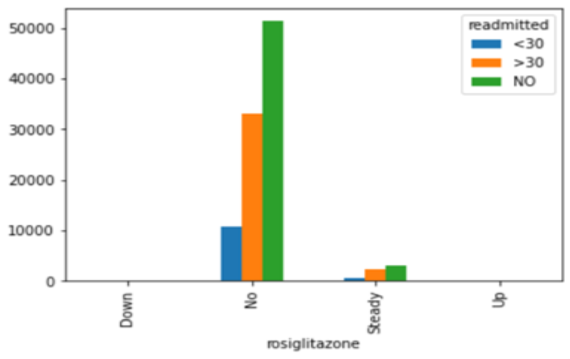
Medications vs readmission: Only Insulin has balanced data

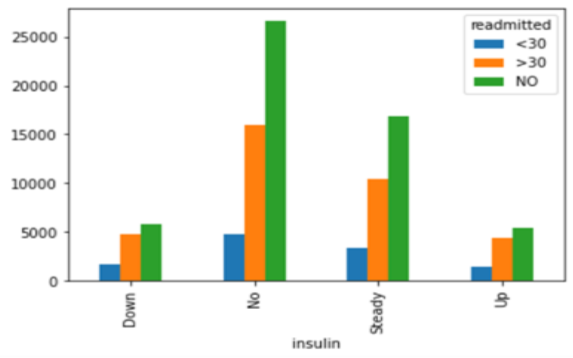


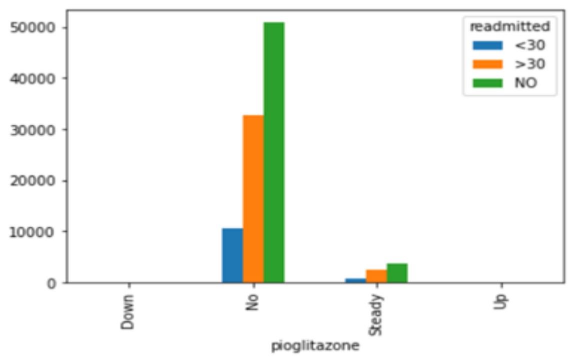




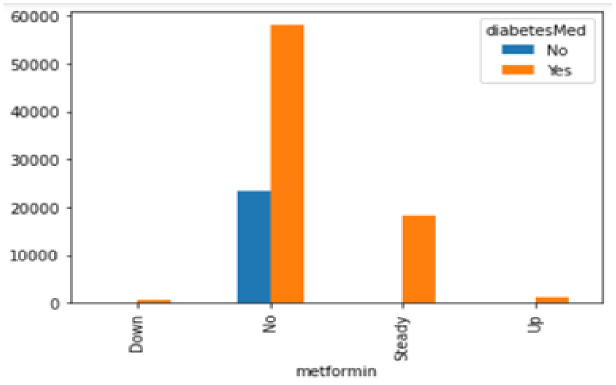


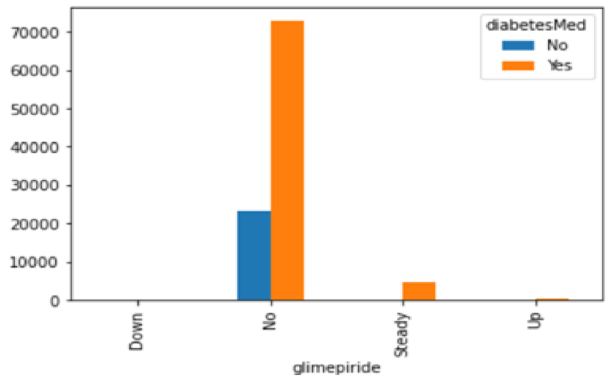


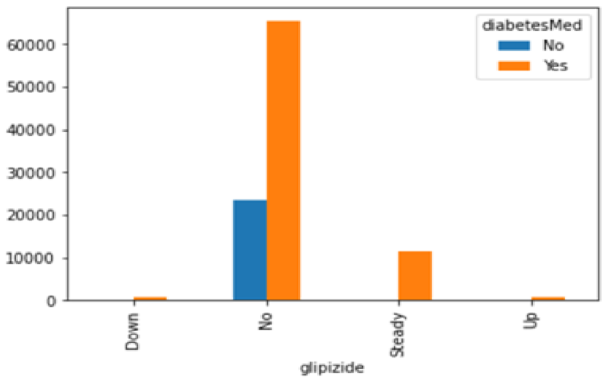


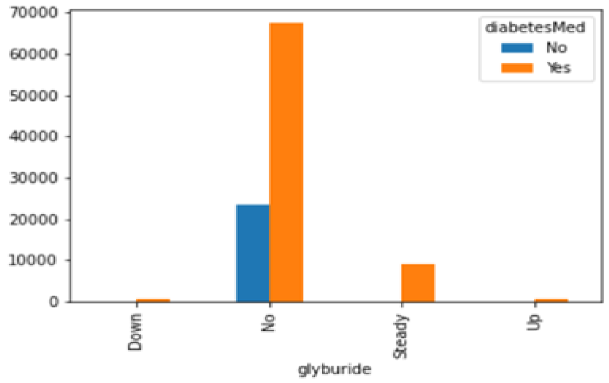


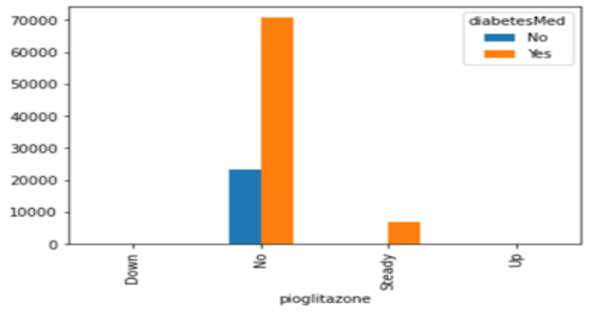
<u>Medications vs Diabetes Meds:</u> Insulin is the only medication having any significant records

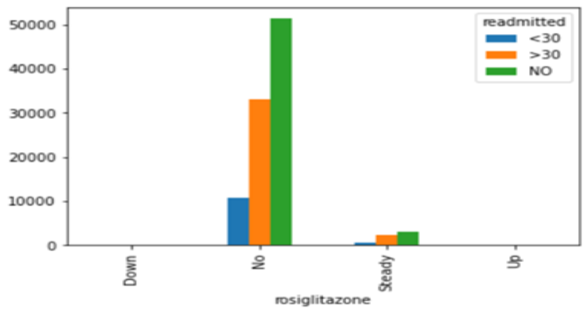


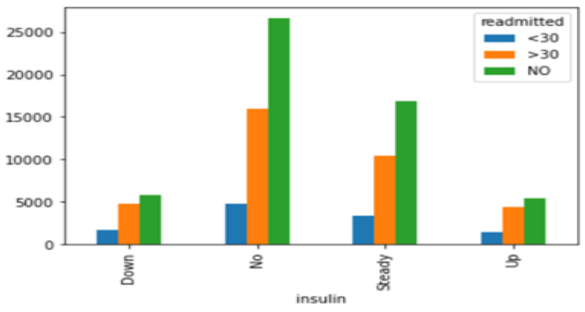








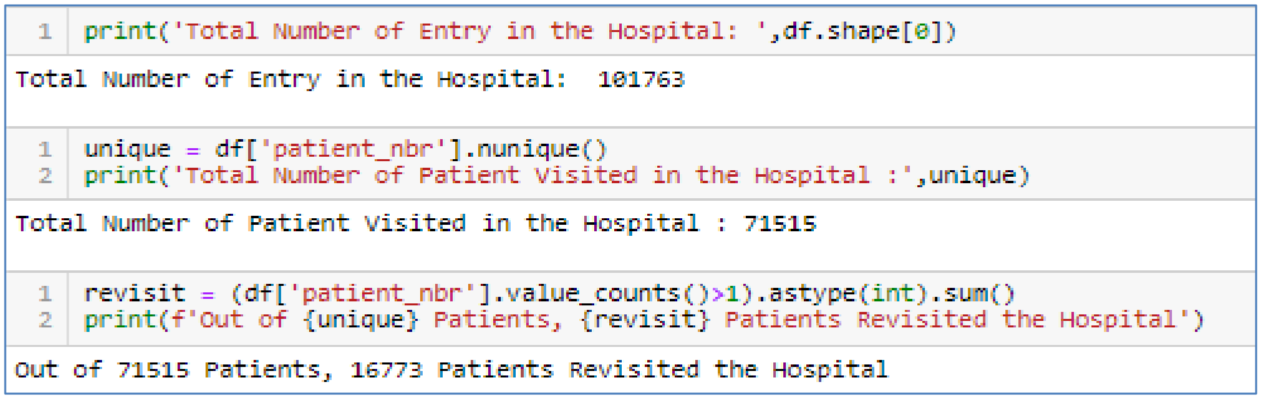




FEATURE ENGINEERING:

Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithms. Feature engineering can be considered as applied machine learning itself.

Patient Number:



```
print('Total Number of Entry in the Hospital: ',df.shape[0])
```

```
Total Number of Entry in the Hospital:  101763
```

```
unique = df['patient_nbr'].nunique()
print('Total Number of Patient Visited in the Hospital :',unique)
```

```
Total Number of Patient Visited in the Hospital : 71515
```

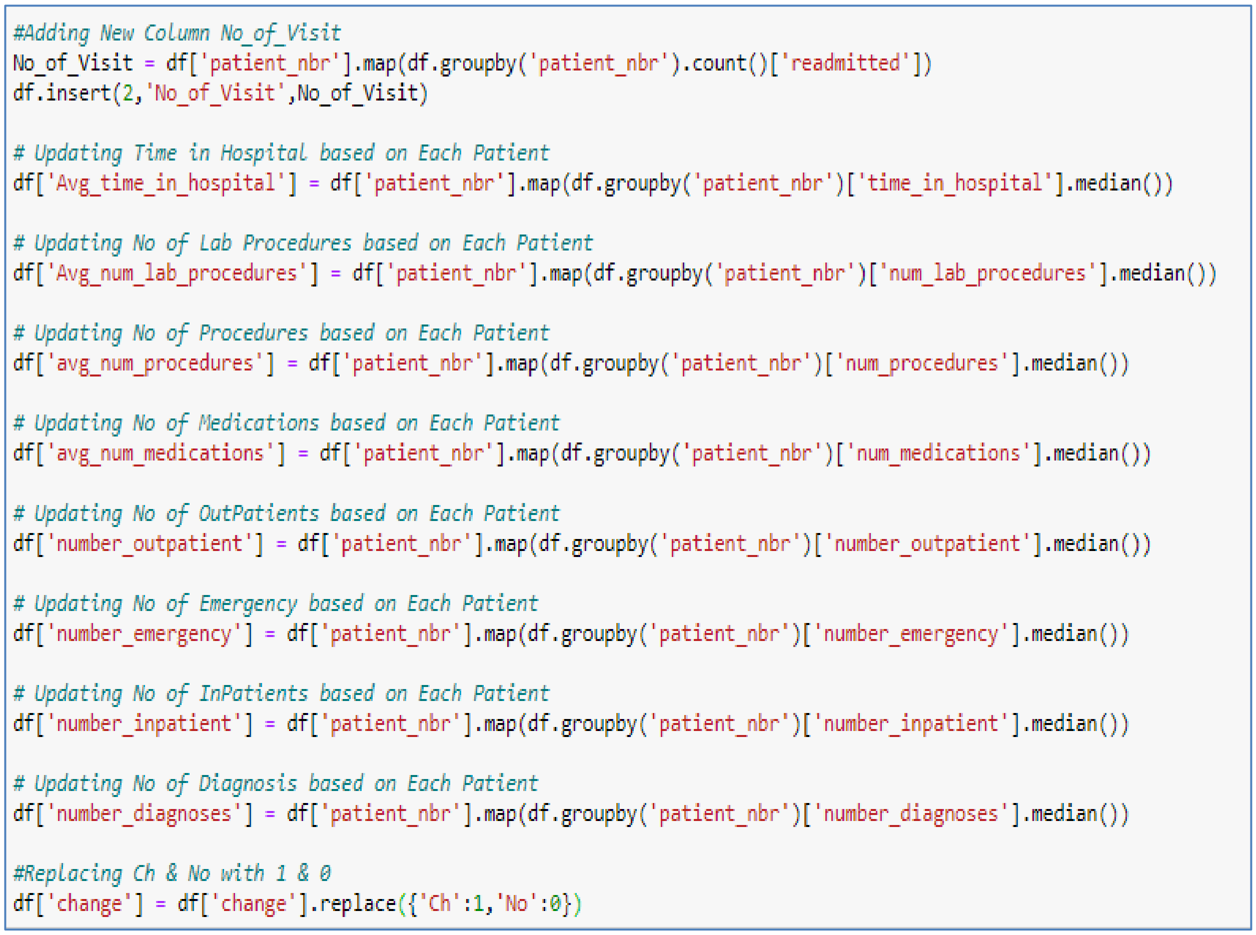
```
revisit = (df['patient_nbr'].value_counts()>1).astype(int).sum()
print(f'Out of {unique} Patients, {revisit} Patients Revisited the Hospital')
```

```
Out of 71515 Patients, 16773 Patients Revisited the Hospital
```

Here we found that 16773 patients revisited hospital. This means a single patient has multiple records. This could lead to duplicates in the records and give faulty predictions towards the target variable. Thus a new feature ‘No_of_Visit’ is created to count the number of visitations by each patient.

[‘time_in_hospital’, ’num_lab_procedures’, ’num_medications’, ‘num_inpatient’, ’num_outpatient’, ‘number_emergency’, ‘number_diagnoses’]

In the above mentioned columns, the median value of each of the columns is imputed after grouping by patient_nbr, in effort to look at the dataset from per patient view, instead of per encounter view.



```
#Adding New Column No_of_Visit
No_of_Visit = df['patient_nbr'].map(df.groupby('patient_nbr').count()['readmitted'])
df.insert(2,'No_of_Visit',No_of_Visit)

# Updating Time in Hospital based on Each Patient
df['Avg_time_in_hospital'] = df['patient_nbr'].map(df.groupby('patient_nbr')['time_in_hospital'].median())

# Updating No of Lab Procedures based on Each Patient
df['Avg_num_lab_procedures'] = df['patient_nbr'].map(df.groupby('patient_nbr')['num_lab_procedures'].median())

# Updating No of Procedures based on Each Patient
df['avg_num_procedures'] = df['patient_nbr'].map(df.groupby('patient_nbr')['num_procedures'].median())

# Updating No of Medications based on Each Patient
df['avg_num_medications'] = df['patient_nbr'].map(df.groupby('patient_nbr')['num_medications'].median())

# Updating No of OutPatients based on Each Patient
df['number_outpatient'] = df['patient_nbr'].map(df.groupby('patient_nbr')['number_outpatient'].median())

# Updating No of Emergency based on Each Patient
df['number_emergency'] = df['patient_nbr'].map(df.groupby('patient_nbr')['number_emergency'].median())

# Updating No of InPatients based on Each Patient
df['number_inpatient'] = df['patient_nbr'].map(df.groupby('patient_nbr')['number_inpatient'].median())

# Updating No of Diagnosis based on Each Patient
df['number_diagnoses'] = df['patient_nbr'].map(df.groupby('patient_nbr')['number_diagnoses'].median())

#Replacing Ch & No with 1 & 0
df['change'] = df['change'].replace({'Ch':1,'No':0})
```

Handling Medical Tests:

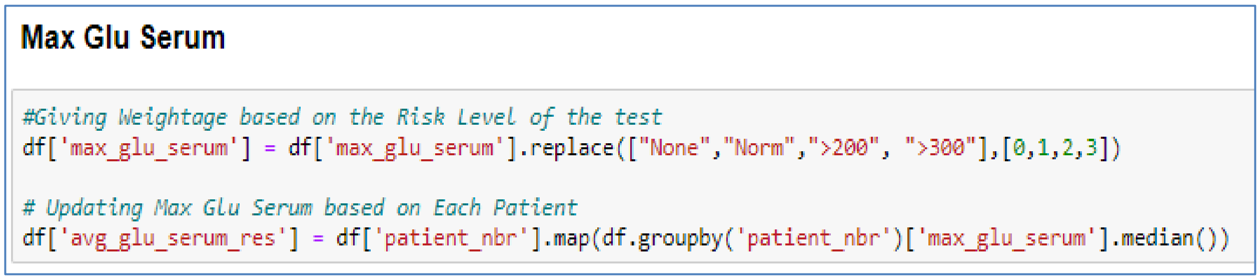
1. Max Glucose Serum Test
2. A1C Result

Max Glu Serum:

- Max Glu Serum test measures the amount of Glucose in the fluid portion of the blood.
- It is measured in milligrams per decilitre.
- Prediabetes indicates the sugar level is high but not that high to have diabetes.

➢ **None**: Test not conducted.
➢ **Norm**: Glucose level is in normal/healthy range.
➢ **>200**: Glucose level indicates prediabetes.
➢ **>300**: Glucose level is high.

Weights have been assigned to the levels of max_glu_serum in the order of risk and grouping by patient number and median function.


Max Glu Serum

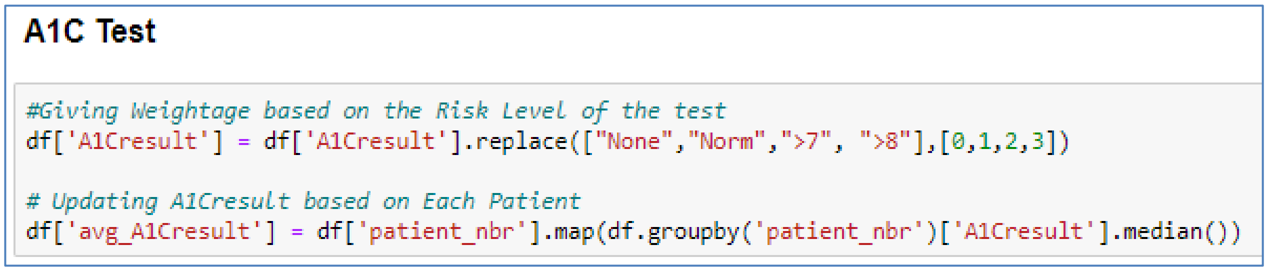
```
#Giving Weightage based on the Risk Level of the test
df['max_glu_serum'] = df['max_glu_serum'].replace(["None","Norm",">200", ">300"],[0,1,2,3])

# Updating Max Glu Serum based on Each Patient
df['avg_glu_serum_res'] = df['patient_nbr'].map(df.groupby('patient_nbr')['max_glu_serum'].median())
```

A1Cresult:

- The A1C test measures the percentage of Red Blood Cells that have sugar-coated hemoglobin.
- Prediabetes indicates the A1C Range is high but not that high to have diabetes.

➢ **None**: Test not conducted.
➢ **Norm**: A1C level is in normal/healthy range.
➢ **>7**: A1C level indicates prediabetes.
➢ **>8**: A1C level is high.

For this column, the weights are assigned based on the levels of risks associated with the test results, from 0-3, 3 being or higher risk, after grouping by patient_nbr and applying median on it as depicted below.



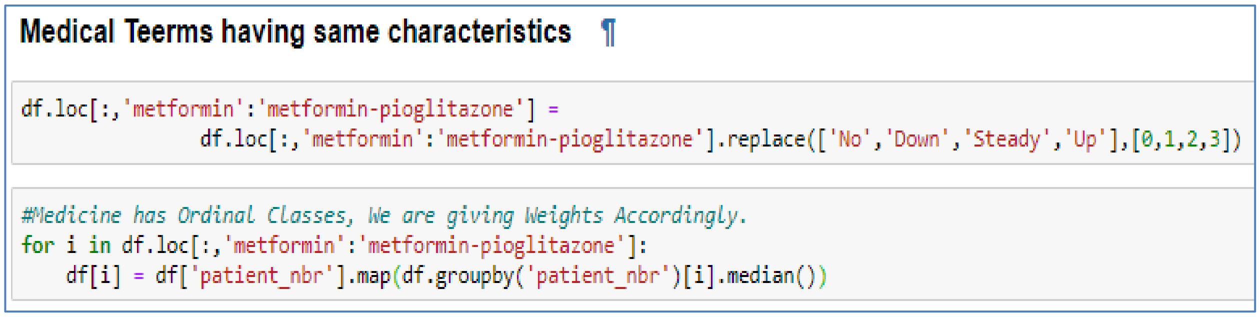
A1C Test

```
#Giving Weightage based on the Risk Level of the test
df['A1Cresult'] = df['A1Cresult'].replace(["None","Norm",">7", ">8"],[0,1,2,3])

# Updating A1Cresult based on Each Patient
df['avg_A1Cresult'] = df['patient_nbr'].map(df.groupby('patient_nbr')['A1Cresult'].median())
```

Drugs prescribed:

Metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, citoglipton, insulin,glyburide-metformin,glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, metformin-pioglitazone.

- The above mentioned medicines all have 4 classes within them, i.e 'No', 'Down', 'Steady', 'Up'. This will be replaced by weights accordingly.



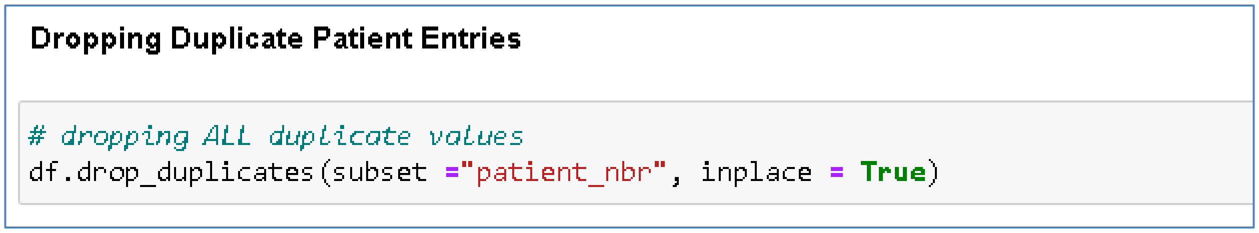
Medical Teerms having same characteristics ¶

```
df.loc[:,'metformin':'metformin-pioglitazone'] =
                df.loc[:,'metformin':'metformin-pioglitazone'].replace(['No','Down','Steady','Up'],[0,1,2,3])
```

```
#Medicine has Ordinal Classes, We are giving Weights Accordingly.
for i in df.loc[:,'metformin':'metformin-pioglitazone']:
    df[i] = df['patient_nbr'].map(df.groupby('patient_nbr')[i].median())
```

Duplicates in Patient entries:

- The duplicate values in patient entries can be dropped as follows:



Dropping Duplicate Patient Entries

```
# dropping ALL duplicate values
df.drop_duplicates(subset ="patient_nbr", inplace = True)
```

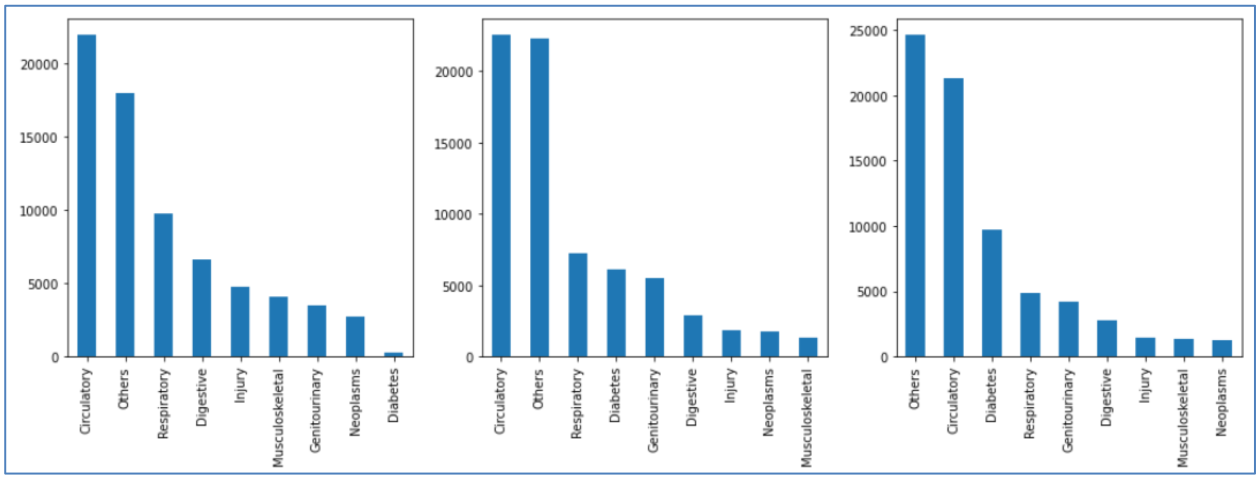
Encoding Techniques :

1. One Hot Encoding

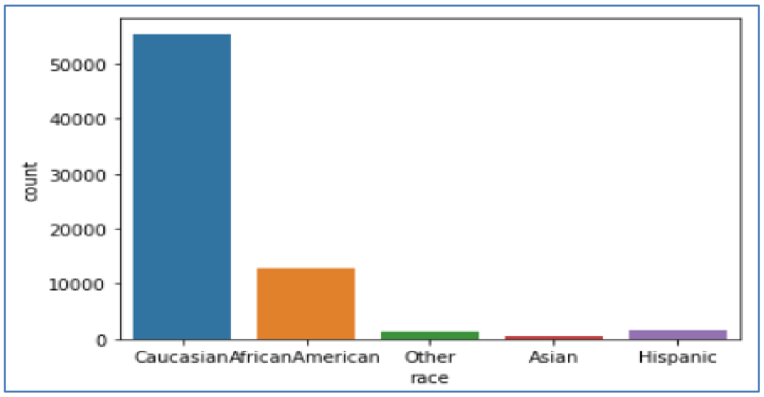
In this method, we map each category to a vector that contains 1 and 0 denoting the presence of the feature or not. The number of vectors depends on the categories which we want to keep. For high cardinality features, this method produces a lot of columns that slows down the learning significantly. There is a buzz between one hot encoding and dummy encoding and when to use one. They are much alike except one hot encoding produces the number of columns equal to the number of categories and dummy producing is one less. This should ultimately be handled by the modeler accordingly in the validation process.

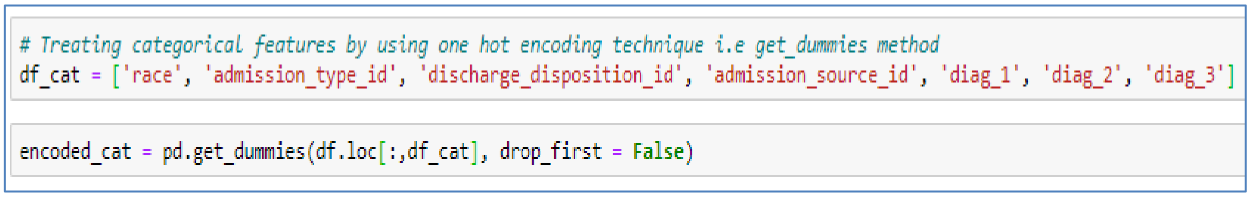
Implementing one hot encoding here to encode the following columns, race, admission_type_id, discharge_disposition_id, admission_source_id, diag_1, diag_2, diag_3, and dropping first columns, as shown below.

Treating Diagnosis Columns with One Hot Encoding:



Treating Race Column with One Hot Encoding:





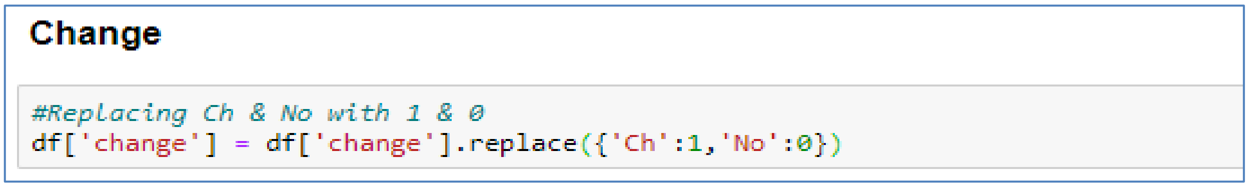
```
# Treating categorical features by using one hot encoding technique i.e get_dummies method
df_cat = ['race', 'admission_type_id', 'discharge_disposition_id', 'admission_source_id', 'diag_1', 'diag_2', 'diag_3']
```

```
encoded_cat = pd.get_dummies(df.loc[:,df_cat], drop_first = False)
```

Replacement by imputation:

Change:

This column has only 2 levels – ‘Ch’ and ‘No’, which are replaced by ‘1’ and ‘0’ respectively.

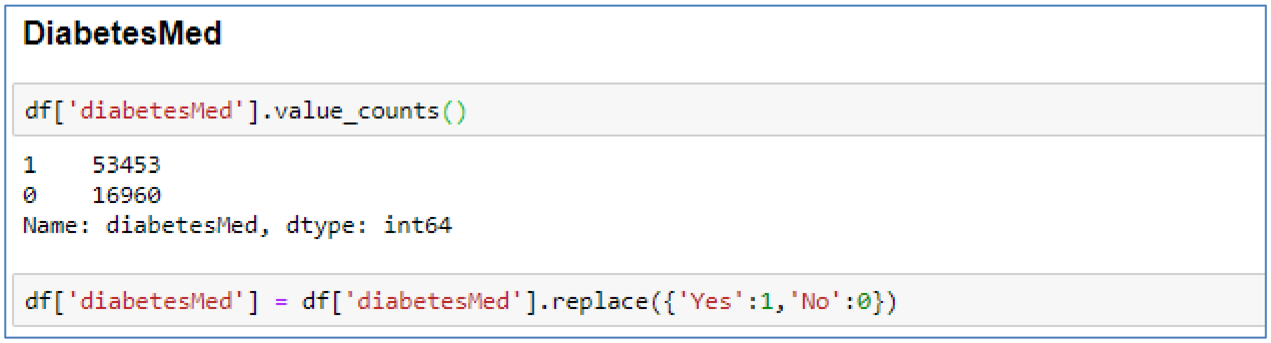


Change

```
#Replacing Ch & No with 1 & 0
df['change'] = df['change'].replace({'Ch':1,'No':0})
```

DiabetesMed:

This column has only 2 levels – ‘Yes’ and ‘No’, which are replaced by ‘1’ and ‘0’ respectively.



DiabetesMed

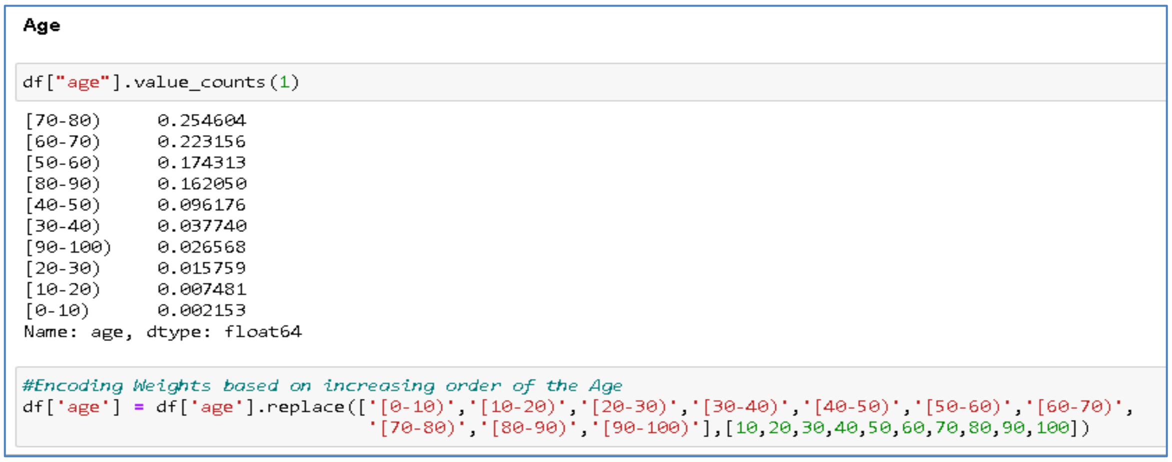
```
df['diabetesMed'].value_counts()
```

```
1    53453
0    16960
Name: diabetesMed, dtype: int64
```

```
df['diabetesMed'] = df['diabetesMed'].replace({'Yes':1,'No':0})
```

Age:

Age column has 10 age brackets, as following [0-10], [10-20], [20-30], [30-40], [40-50], [50-60], [60-70], [70-80], [80-90] and [90-100] which are replaced by integer values 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 respectively as below.



Age

```
df["age"].value_counts(1)
```

```
[70-80)      0.254604
[60-70)      0.223156
[50-60)      0.174313
[80-90)      0.162050
[40-50)      0.096176
[30-40)      0.037740
[90-100)     0.026568
[20-30)      0.015759
[10-20)      0.007481
[0-10)       0.002153
Name: age, dtype: float64
```

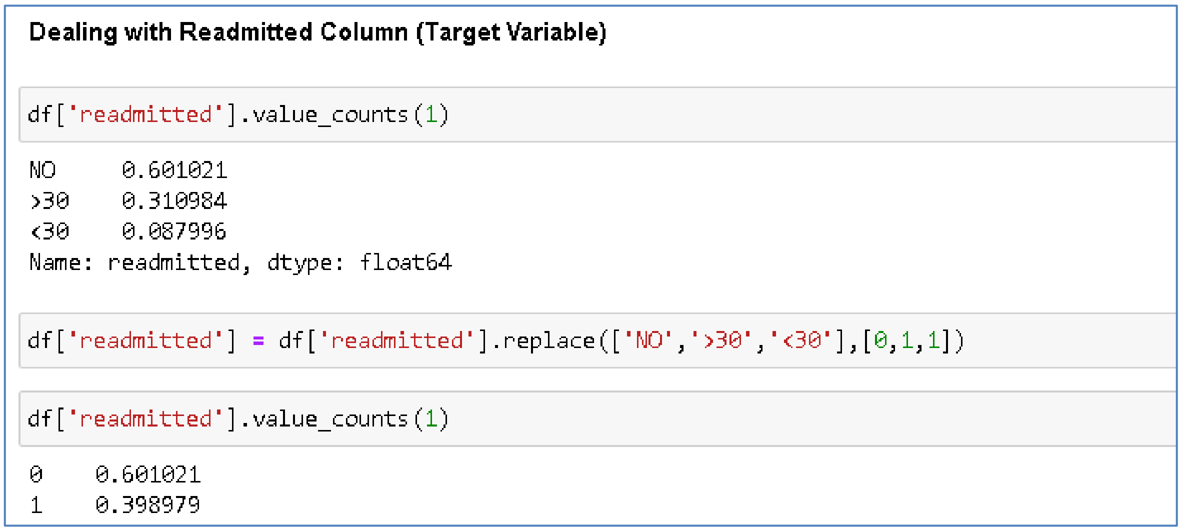
```
#Encoding Weights based on increasing order of the Age
df['age'] = df['age'].replace(['[0-10)','[10-20)','[20-30)','[30-40)','[40-50)','[50-60)','[60-70)',
                               '[70-80)','[80-90)','[90-100)'],[10,20,30,40,50,60,70,80,90,100])
```

Target Variable: Readmitted

The target variable has 3 levels,

- 'NO' – indicating patient was not readmitted
- '>30' – indicating patient was readmitted after 30 days
- '<30' - indicating patient was readmitted before 30 days

Here we impute, patient not being readmitted as '0' and the other two cases, of patient readmission as '1', irrespective of being readmitted before or after 30 days from being discharged from hospital from the first encounter.



Dealing with Readmitted Column (Target Variable)

```
df['readmitted'].value_counts(1)
```

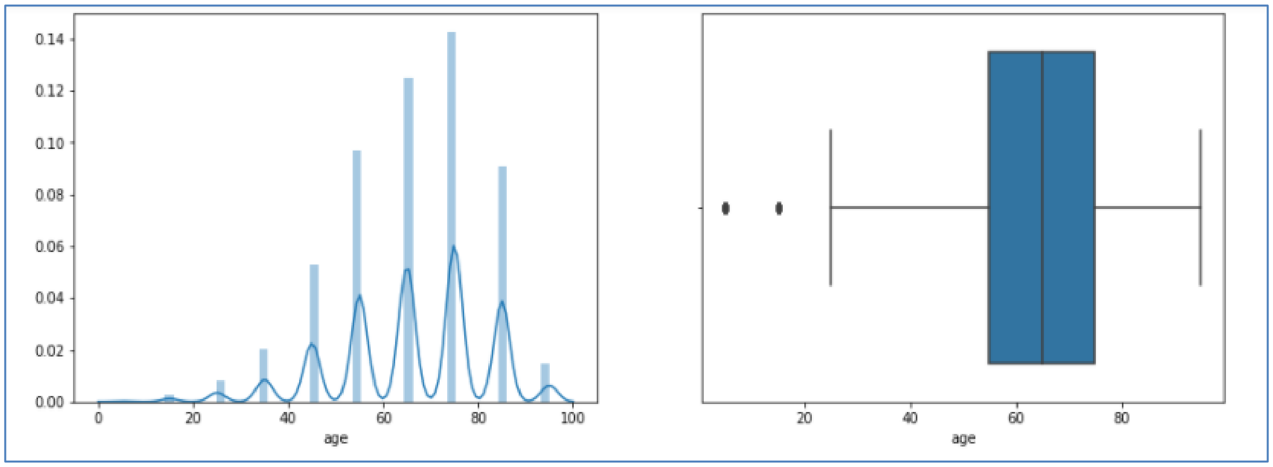
```
NO     0.601021
>30    0.310984
<30    0.087996
Name: readmitted, dtype: float64
```

```
df['readmitted'] = df['readmitted'].replace(['NO','>30','<30'],[0,1,1])
```

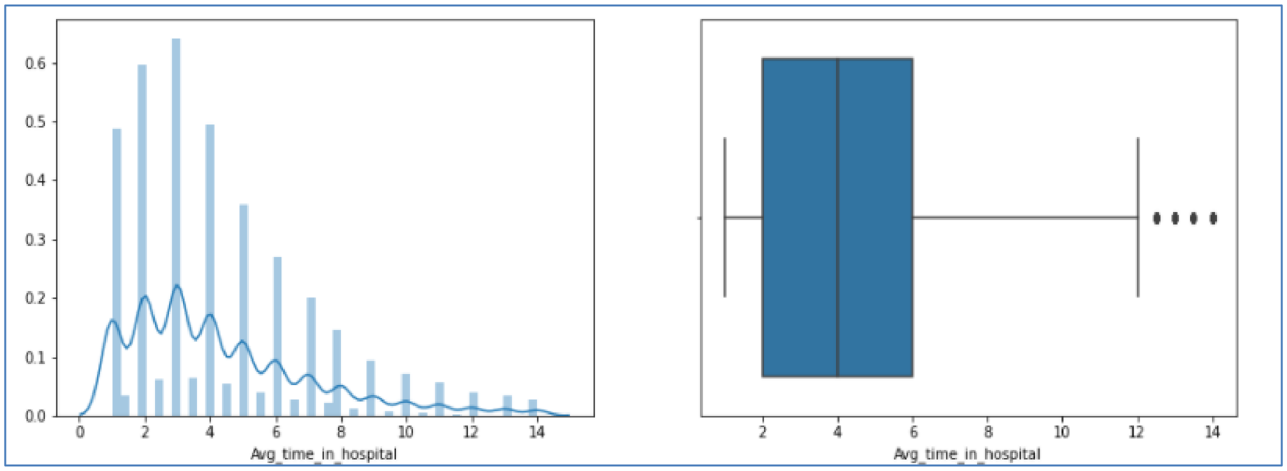
```
df['readmitted'].value_counts(1)
```

```
0    0.601021
1    0.398979
```

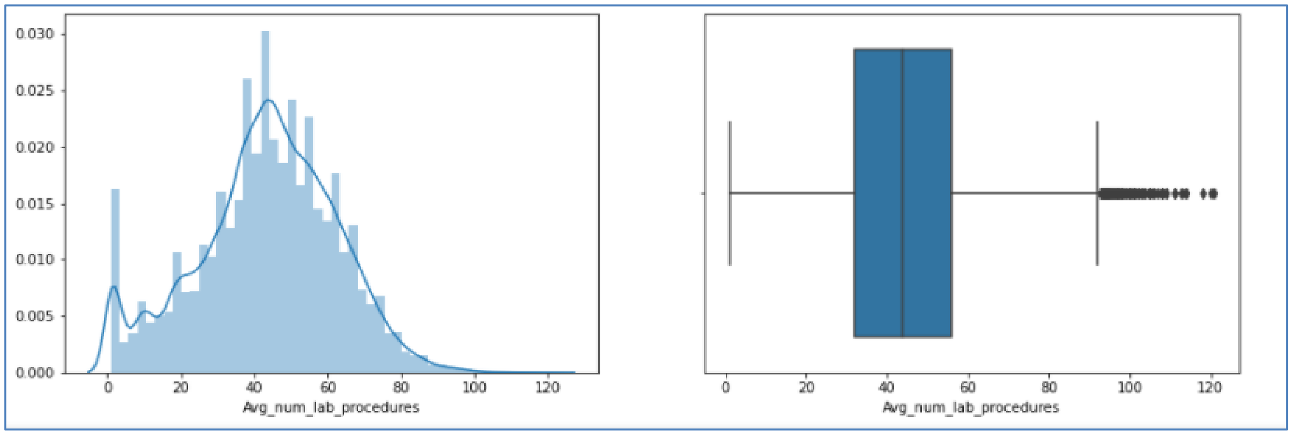
Outlier Treatment in Numerical Features:



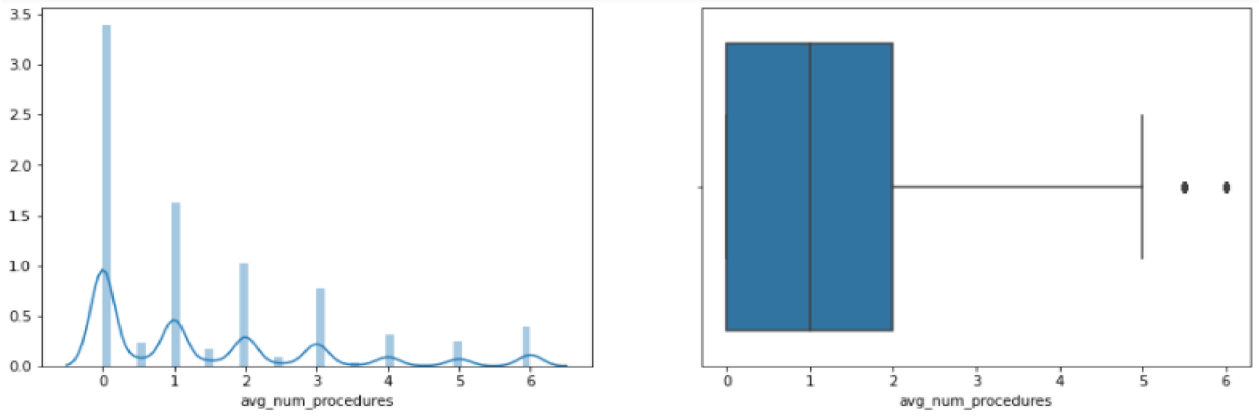
Age



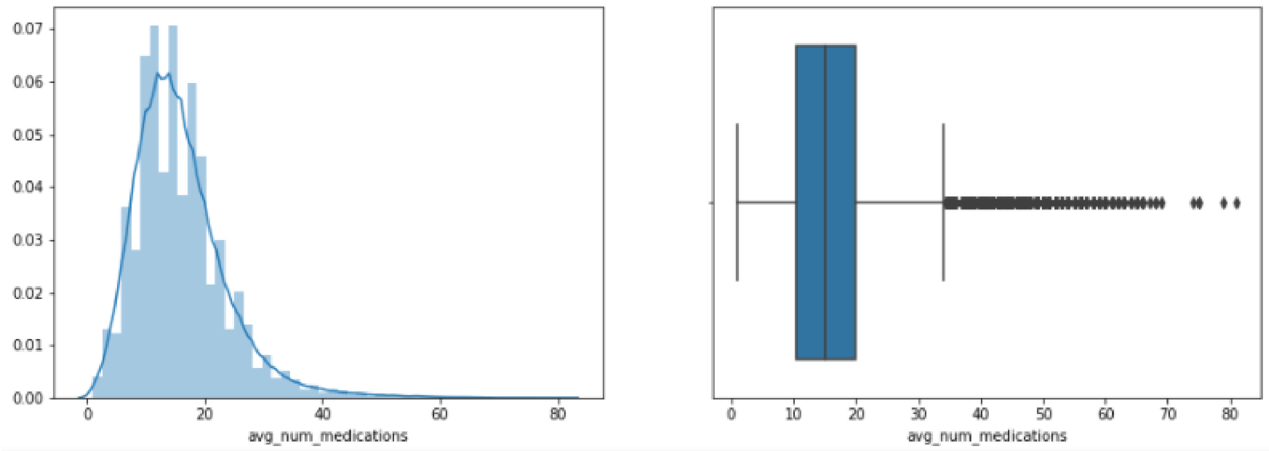
Average Time in Hospital



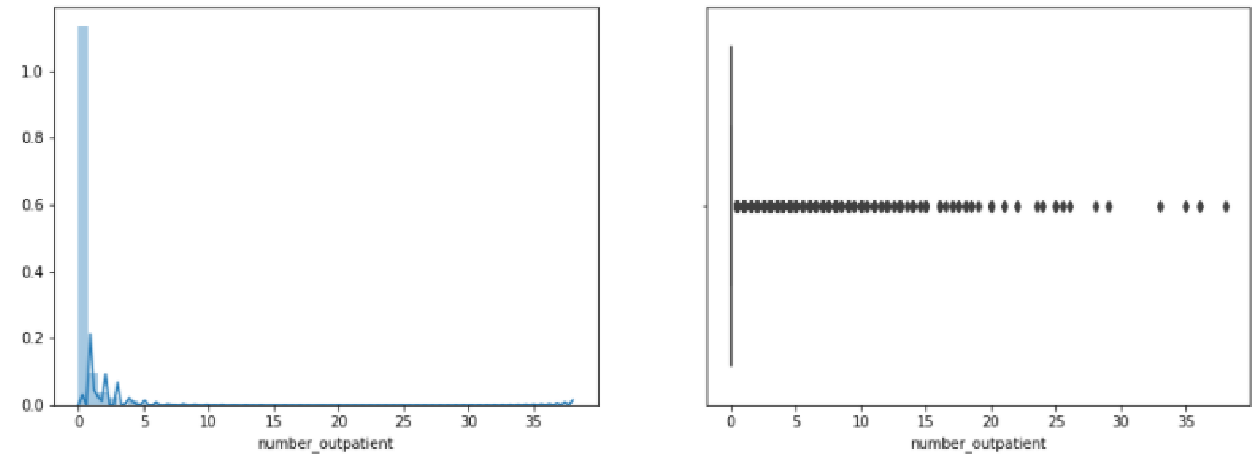
Average Number of Lab Procedures



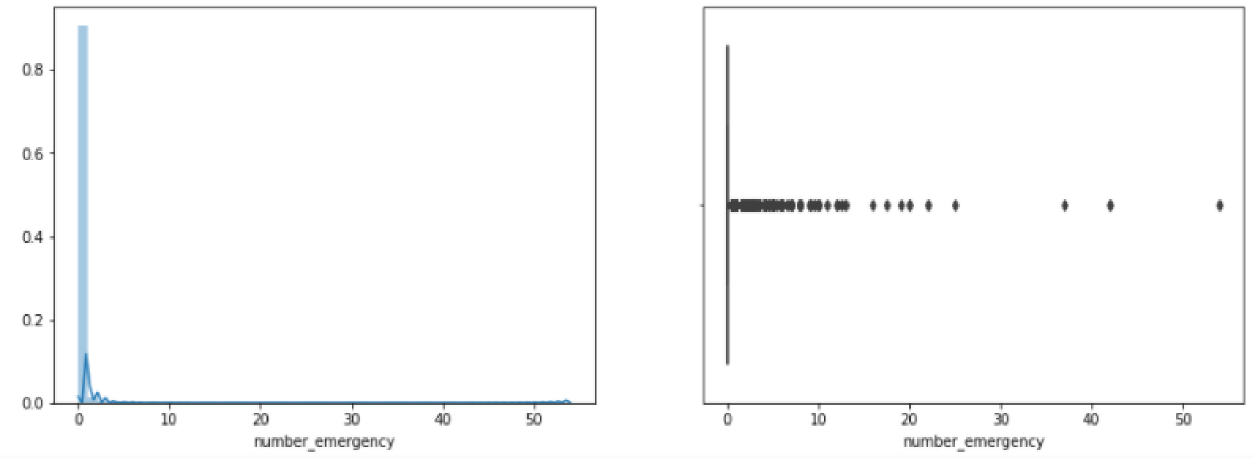
Average Number of Procedures



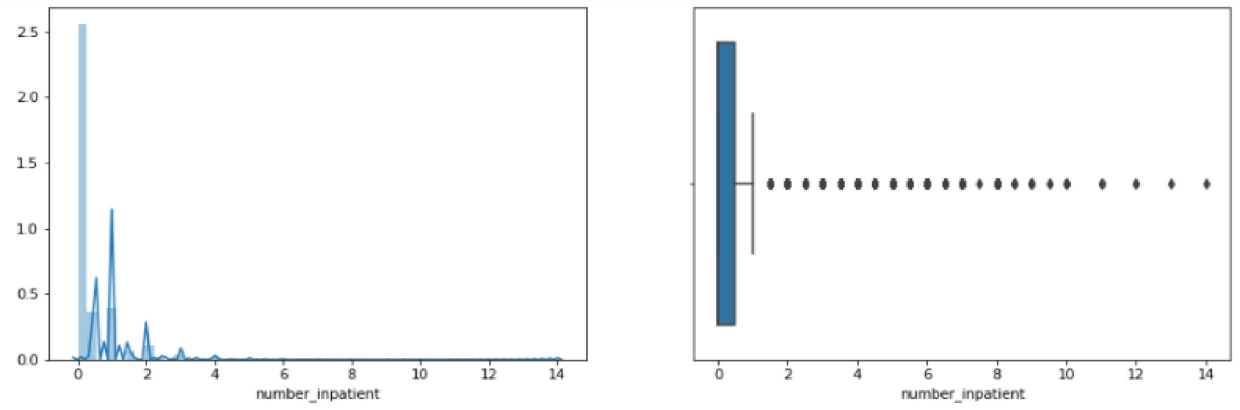
Average Number of Medications



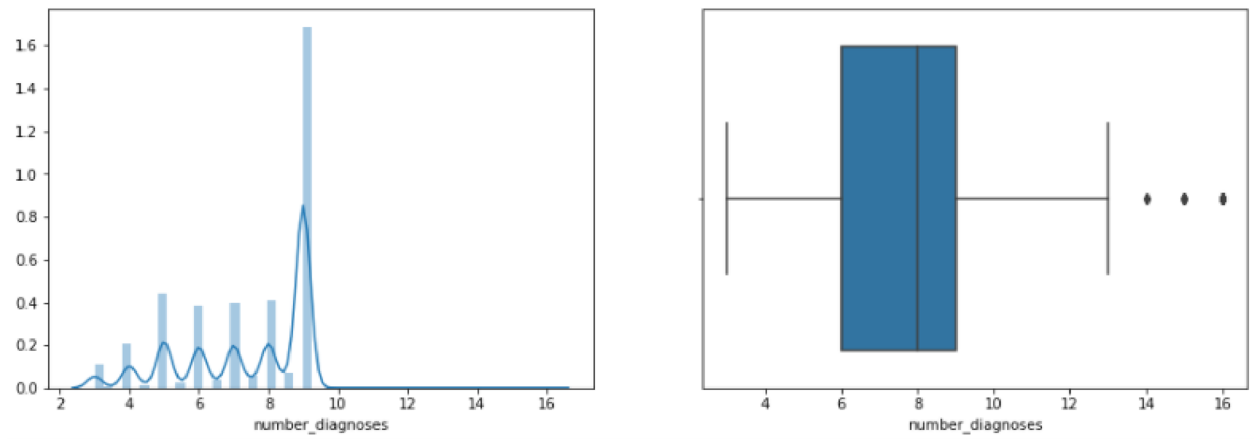
Average Number of Outpatients



Number of Emergency



Number of Inpatient



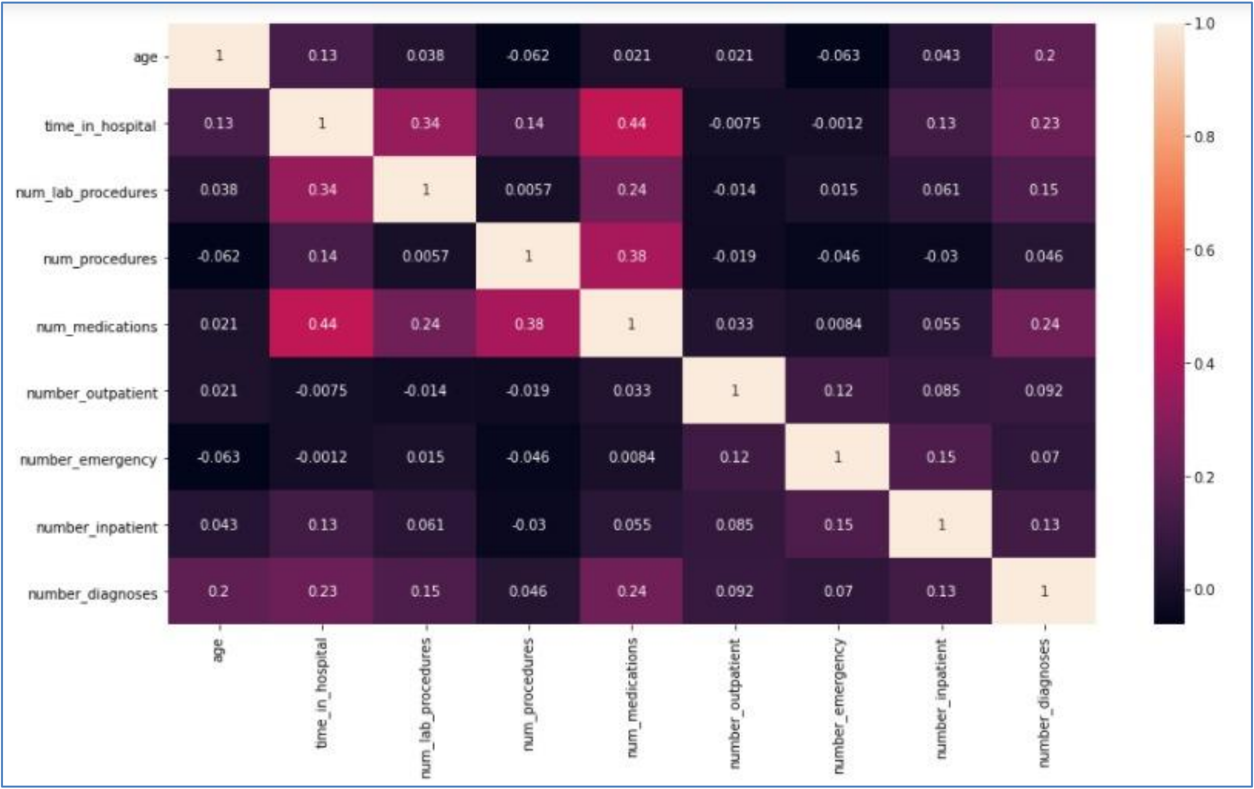
Number of Diagnosis

All the Outliers observed above have been treated with the Inter Quantile Range & using transformation [Log, Squareroot & Power Transformer]

All the columns in the dataset were divided into 2 groups, i.e. Categorical and numerical values. Categorical variables are tested with Chisquare to test the level of significance for all the variables with the Target column.

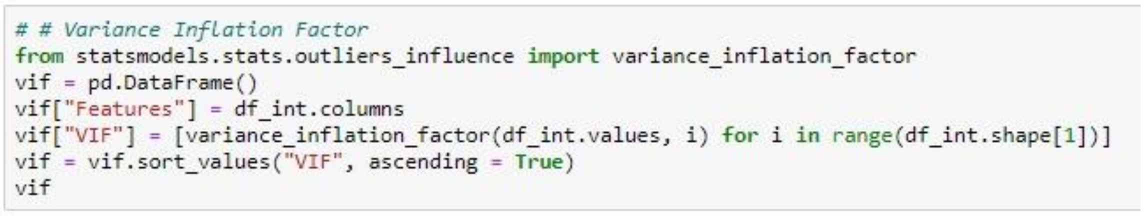
Correlation between Independent Numerical Features:

We have found the correlation between Independent Numerical features by using Heatmap.

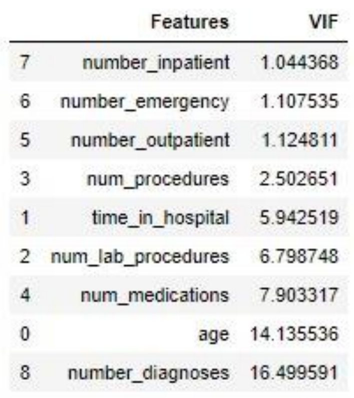


- We have found that there is no Correlation between any Independent Features.
- Not a Single Feature is having Correlation value more than 0.5.
- However, we are cross checking using Variance Inflation Factor.

Variance Inflation Factor:

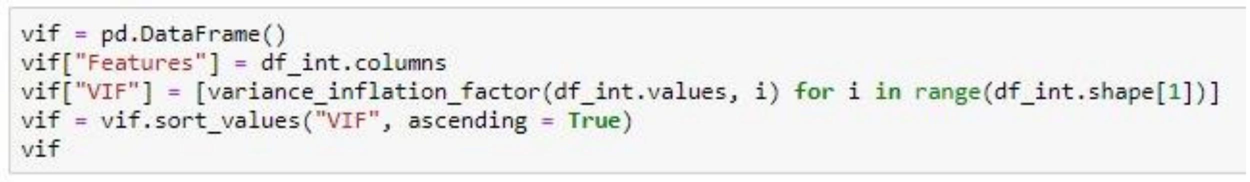


```
# # Variance Inflation Factor
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif["Features"] = df_int.columns
vif["VIF"] = [variance_inflation_factor(df_int.values, i) for i in range(df_int.shape[1])]
vif = vif.sort_values("VIF", ascending = True)
vif
```

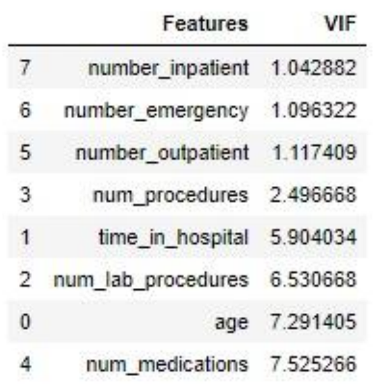


	Features	VIF
7	number_inpatient	1.044368
6	number_emergency	1.107535
5	number_outpatient	1.124811
3	num_procedures	2.502651
1	time_in_hospital	5.942519
2	num_lab_procedures	6.798748
4	num_medications	7.903317
0	age	14.135536
8	number_diagnoses	16.499591

- Based on VIF Scores, we observe Number of Diagnosis Feature is having Multicollinearity characteristics.
- Thus In order to remove the Multicollinearity from the data, we have dropped the ‘Number of Diagnosis’ Feature.



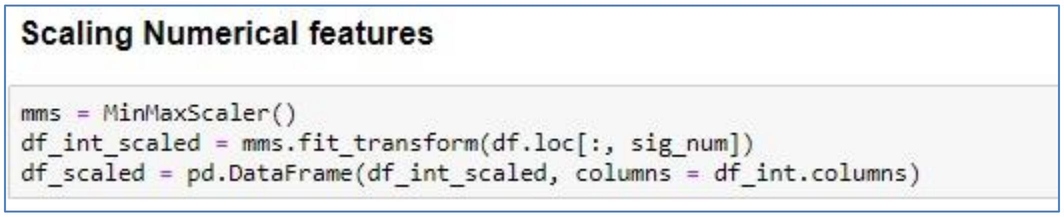
```
vif = pd.DataFrame()
vif["Features"] = df_int.columns
vif["VIF"] = [variance_inflation_factor(df_int.values, i) for i in range(df_int.shape[1])]
vif = vif.sort_values("VIF", ascending = True)
vif
```



	Features	VIF
7	number_inpatient	1.042882
6	number_emergency	1.096322
5	number_outpatient	1.117409
3	num_procedures	2.496668
1	time_in_hospital	5.904034
2	num_lab_procedures	6.530668
0	age	7.291405
4	num_medications	7.525266

- After Removal of Feature ‘Number of Diagnosis’, We can observe Age and Avg No of Meds are having moderate multicollinearity

Feature Scaling: Min Max Scaler used for Scaling Numerical Independent Features.

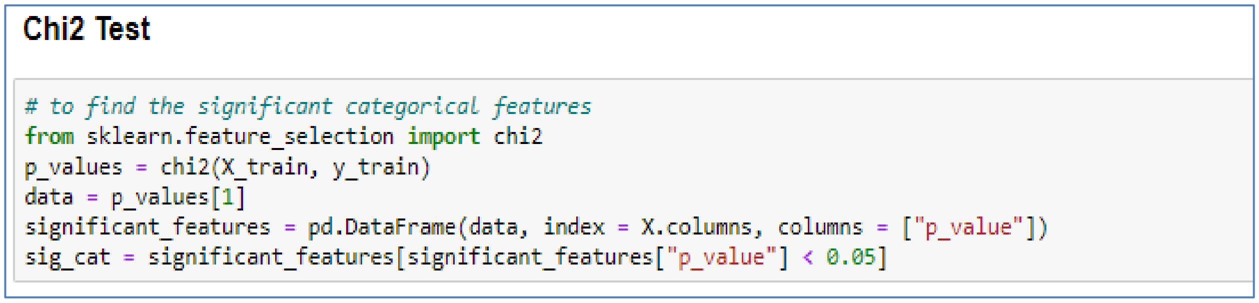


Scaling Numerical features

```
mms = MinMaxScaler()
df_int_scaled = mms.fit_transform(df.loc[:, sig_num])
df_scaled = pd.DataFrame(df_int_scaled, columns = df_int.columns)
```

Statistical Analysis: Chi Square Test

A chi-square test for independence compares two variables in a contingency table to see if they are related. In a more general sense, it tests to see whether distributions of categorical variables differ from each another. A very small chi square test statistic means that your observed data fits your expected data extremely well. In other words, there is a relationship. A very large chi square test statistic means that the data does not fit very well. In other words, there isn't a relationship.



Chi2 Test

```
# to find the significant categorical features
from sklearn.feature_selection import chi2
p_values = chi2(X_train, y_train)
data = p_values[1]
significant_features = pd.DataFrame(data, index = X.columns, columns = ["p_value"])
sig_cat = significant_features[significant_features["p_value"] < 0.05]
```

Table of variables and their significance:

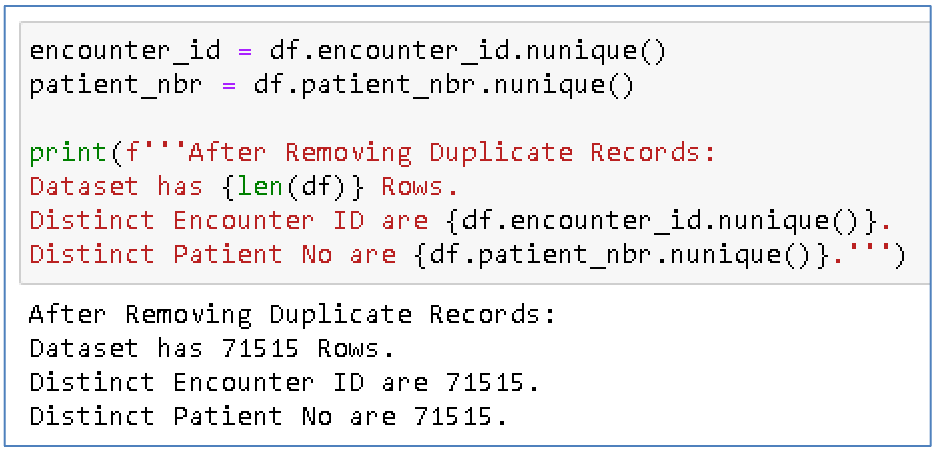
Sr. No.	Features	P value	Significance
1	No of Visit	0	Significant
2	Gender	9.8e-3	Significant
3	Metformin	6.85e-18	Significant
4	Repaglinide	3.77e-11	Significant
5	Nateglinide	9.24e-11	Significant
6	Chlorpropamide	0.15	Insignificant
7	Glimepiride	0.612	Insignificant
8	Acetohexamide	0.0843	Insignificant
9	Glipizide	6.41e-9	Significant
10	Glyburide	0.706	Insignificant

11	Tolbutamide	0.696	Insignificant
12	Pioglitazone	1.957e-4	Significant
13	Rosiglitaone	0.0795	Insignificant
14	Acarbose	0.0203	Insignificant
15	Miglitol	7.42e-4	Significant
16	Tolazamide	0.71	Insignificant
17	Insulin	2.26e-18	Significant
18	Glyburide-metformin	0.87	Insignificant
19	Glipizide-metformin	0.48	Insignificant
20	Metformin-rosiglitazone	0.101	Insignificant
21	Change	1.09e-12	Significant
22	DiabetesMed	1.21e-13	Significant
23	max_glu_serum	1.79e-9	Significant
24	A1Cresult	4.06e-44	Significant
25	race_AfricanAmerican	0.0150	Significant
26	race_Asian	7.87e-5	Significant
27	race_Caucasian	0.0112	Significant
28	race_Hispanic	0.0243	Significant
29	race_Other	0.00617	Significant
30	admission_type_id_Elective	4.58e-23	Significant
31	admission_type_id_Emergency	0.063	Insignificant
32	admission_type_id_Others	3.36e-20	Significant
33	admission_type_id_Urgent	0.91	Insignificant
34	discharge_disposition_id_Discharged	3.64e-4	Significant
35	discharge_disposition_id_Others	1.37e-70	Significant
36	discharge_disposition_id_Transfer	5.56e-42	Significant
37	admission_source_id_Emergency Room	2.06e-16	Significant
38	admission_source_id_Referral	8.622e-9	Significant

39	admission_source_id_Transfer	6.02e-14	Significant
40	diag_1_Circulatory	3.06e-05	Significant
41	diag_1_Diabetes	8.25e-06	Significant
42	diag_1_Digestive	3.019e-01	Insignificant
43	diag_1_Genitourinary	4.817e-03	Significant
44	diag_1_Injury	5.661e-01	Insignificant
45	diag_1_Musculoskeletal	4.29e-06	Significant
46	diag_1_Neoplasms	4.12e-20	Significant
47	diag_1_Other Symptoms	0.101	Insignificant
48	diag_1_Respiratory	2.1279e-03	Significant
49	diag_2_Circulatory	3.6726e-09	Significant
50	diag_2_Diabetes	1.7343e-15	Significant
51	diag_2_Digestive	4.5059e-03	Significant
52	diag_2_Genitourinary	7.2923e-01	Insignificant
53	diag_2_Injury	2.1214e-04	Significant
54	diag_2_Musculoskeletal	3.0545e-02	Significant
55	diag_2_Neoplasms	2.5140e-01	Insignificant
56	diag_2_Other Symptoms	7.9210e-01	Insignificant
57	diag_2_Respiratory	1.1602e-01	Insignificant
58	diag_3_Circulatory	3.2880e-03	Significant
59	diag_3_Diabetes	2.2536e-17	Significant
60	diag_3_Digestive	4.1838e-01	Insignificant
61	diag_3_Genitourinary	2.7963e-03	Significant
62	diag_3_Injury	5.5685e-04	Significant
63	diag_3_Musculoskeletal	9.5222e-01	Insignificant
64	diag_3_Neoplasms	8.2863e-02	Insignificant
65	diag_3_Other Symptoms	5.0163e-01	Insignificant
66	diag_3_Respiratory	1.2843e-04	Significant

- We drop all the Insignificant Features from the above Table.
- As it is Statistically Insignificant.

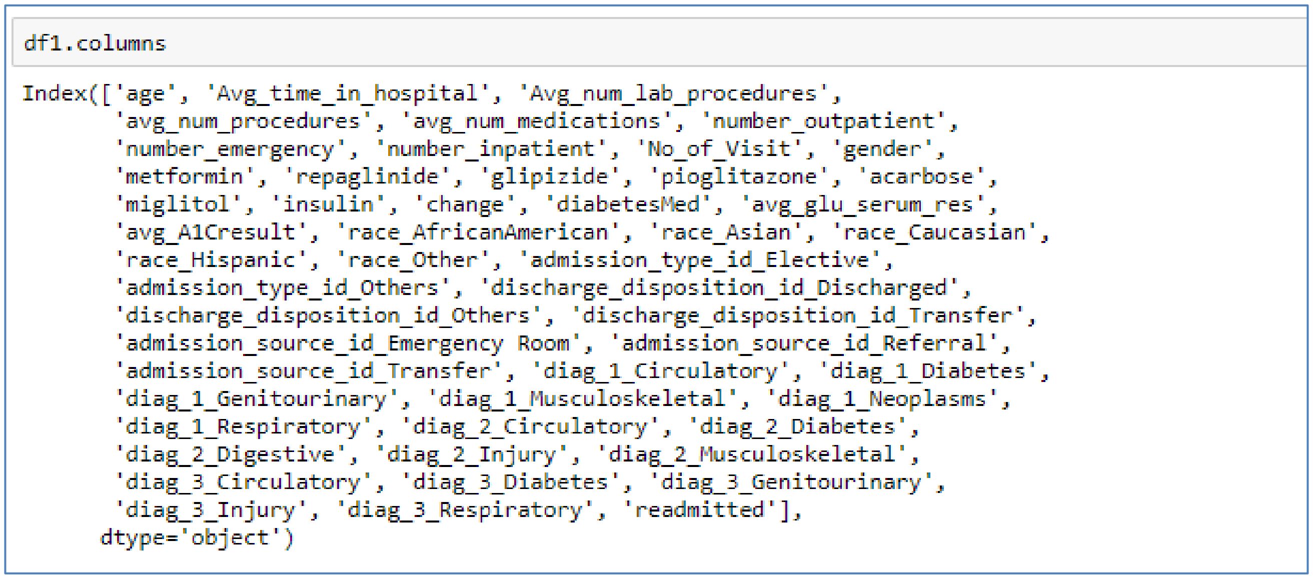
<u>**Encounter ID and Patient Number:**</u>
Both these columns have unique records after removing all duplicates, in the above steps. So, both these variables will not contribute towards building a robust model. After dropping these, there are now 71515 rows and 38 columns left on which model will be built and tested for classification metrics.



```
encounter_id = df.encounter_id.nunique()
patient_nbr = df.patient_nbr.nunique()

print(f'''After Removing Duplicate Records:
Dataset has {len(df)} Rows.
Distinct Encounter ID are {df.encounter_id.nunique()}.
Distinct Patient No are {df.patient_nbr.nunique()}.''')
```

```
After Removing Duplicate Records:
Dataset has 71515 Rows.
Distinct Encounter ID are 71515.
Distinct Patient No are 71515.
```

After Dropping All the Above Features, Data is ready for Modelling.
Below mentioned columns are the one, which will be used for building models.



```
df1.columns
```

```
Index(['age', 'Avg_time_in_hospital', 'Avg_num_lab_procedures',
       'avg_num_procedures', 'avg_num_medications', 'number_outpatient',
       'number_emergency', 'number_inpatient', 'No_of_Visit', 'gender',
       'metformin', 'repaglinide', 'glipizide', 'pioglitazone', 'acarbose',
       'miglitol', 'insulin', 'change', 'diabetesMed', 'avg_glu_serum_res',
       'avg_A1Cresult', 'race_AfricanAmerican', 'race_Asian', 'race_Caucasian',
       'race_Hispanic', 'race_Other', 'admission_type_id_Elective',
       'admission_type_id_Others', 'discharge_disposition_id_Discharged',
       'discharge_disposition_id_Others', 'discharge_disposition_id_Transfer',
       'admission_source_id_Emergency Room', 'admission_source_id_Referral',
       'admission_source_id_Transfer', 'diag_1_Circulatory', 'diag_1_Diabetes',
       'diag_1_Genitourinary', 'diag_1_Musculoskeletal', 'diag_1_Neoplasms',
       'diag_1_Respiratory', 'diag_2_Circulatory', 'diag_2_Diabetes',
       'diag_2_Digestive', 'diag_2_Injury', 'diag_2_Musculoskeletal',
       'diag_3_Circulatory', 'diag_3_Diabetes', 'diag_3_Genitourinary',
       'diag_3_Injury', 'diag_3_Respiratory', 'readmitted'],
      dtype='object')
```

MODEL BUILDING:

For model building we will proceed with a split of 70:30 in the train-test data.

BASE MODEL:

Logistic Regression:

Logistic regression is a transformation of the linear regression model that allows us to probabilistically model binary variables. It is also known as a generalized linear model that uses a logit-link.

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

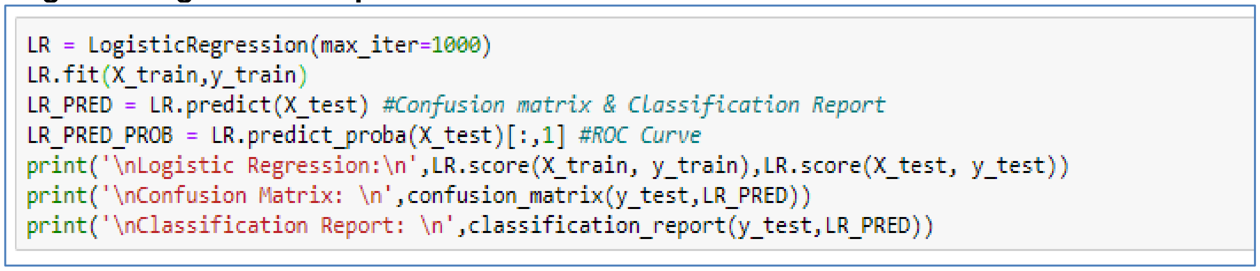
Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.

The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be 1, but this seldom is the case. In case of a Precision-Recall trade-off we use the following arguments to decide upon the thresold:

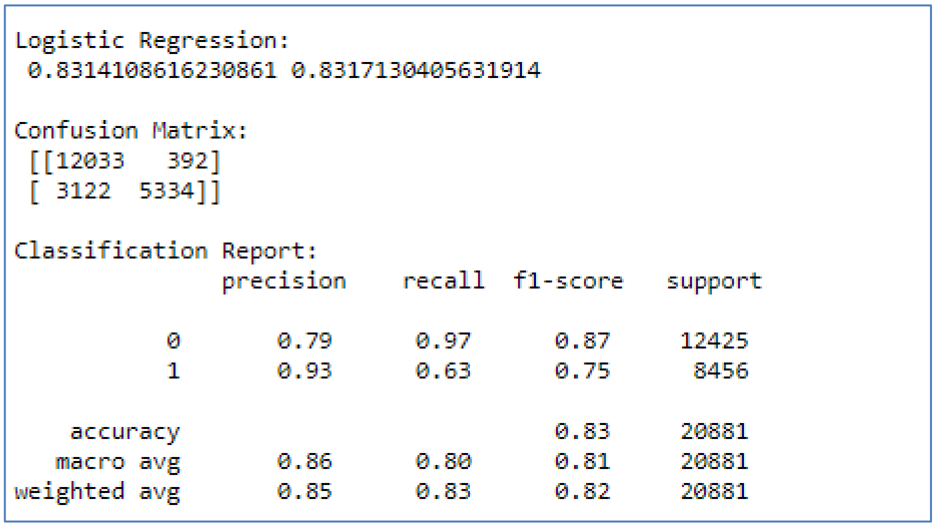
1. **Low Precision/High Recall:** In applications where we want to reduce the number of false negatives without necessarily reducing the number false positives, we choose a decision value which has a low value of Precision or high value of Recall. For example, in a cancer diagnosis application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because, the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2. **High Precision/Low Recall:** In applications where we want to reduce the number of false positives without necessarily reducing the number false negatives, we choose a decision value which has a high value of Precision or low value of Recall. For example, if we are classifying customers whether they will react positively or negatively to a personalised advertisement, we want to be absolutely sure that the customer will react positively to the advertisemnt because otherwise, a negative reaction can cause a loss potential sales from the customer.

Logistic Regression Implimentation:



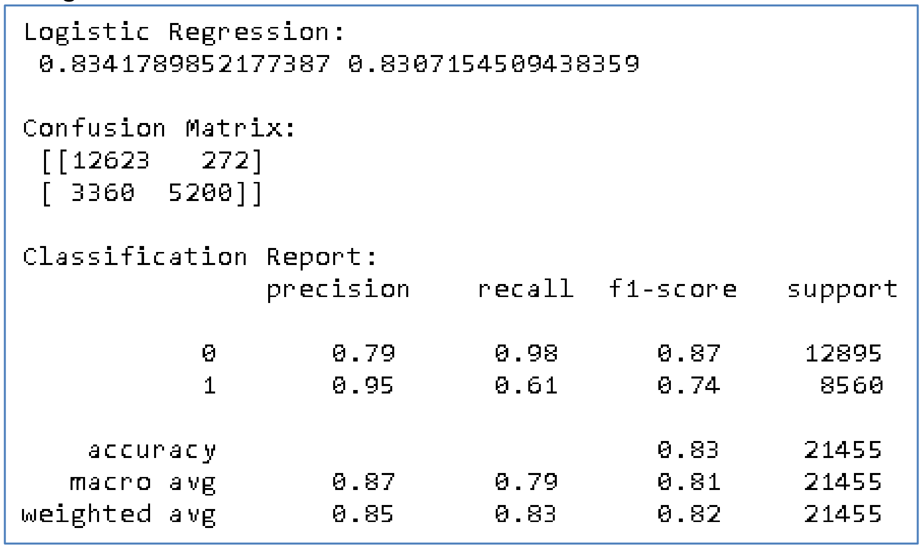
```
LR = LogisticRegression(max_iter=1000)
LR.fit(X_train,y_train)
LR_PRED = LR.predict(X_test) #Confusion matrix & Classification Report
LR_PRED_PROB = LR.predict_proba(X_test)[:,1] #ROC Curve
print('\nLogistic Regression:\n',LR.score(X_train, y_train),LR.score(X_test, y_test))
print('\nConfusion Matrix: \n',confusion_matrix(y_test,LR_PRED))
print('\nClassification Report: \n',classification_report(y_test,LR_PRED))
```



```
Logistic Regression:
 0.8314108616230861 0.8317130405631914

Confusion Matrix:
 [[12033   392]
 [ 3122  5334]]

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.97      0.87     12425
           1       0.93      0.63      0.75      8456

    accuracy                           0.83     20881
   macro avg       0.86      0.80      0.81     20881
weighted avg       0.85      0.83      0.82     20881
```

Hyper Parameter Tuning:

- We tried applying solver, but no significant improvement observed.
- We changed max iter from 100 to 1000



```
Logistic Regression:
 0.8341789852177387 0.8307154509438359

Confusion Matrix:
 [[12623   272]
 [ 3360  5200]]

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.98      0.87     12895
           1       0.95      0.61      0.74      8560

    accuracy                           0.83     21455
   macro avg       0.87      0.79      0.81     21455
weighted avg       0.85      0.83      0.82     21455
```

Advantages:

- Logistic regression is easier to implement, interpret, and very efficient to train.
- It makes no assumptions about distributions of classes in feature space.
- It is very fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- It can interpret model coefficients as indicators of feature importance.

<u>Naïve Bayes:</u>

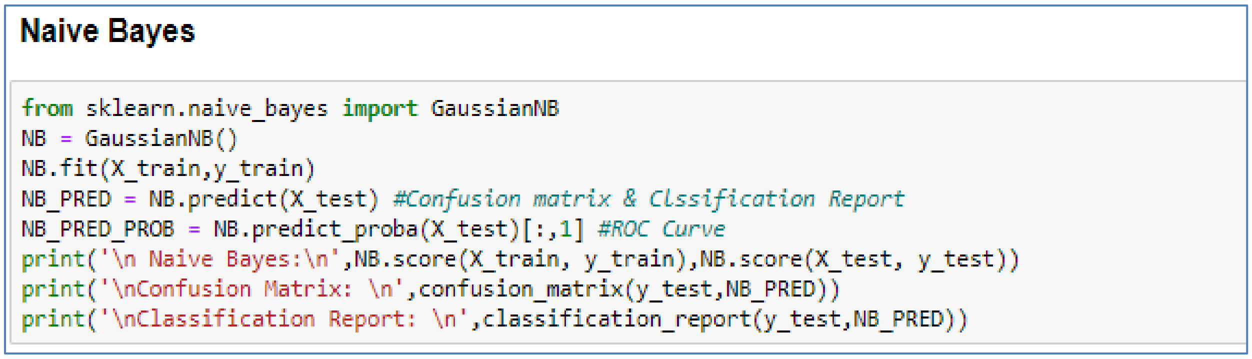
Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.
Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- P(h): the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h.
- P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.
- P(D|h): the probability of data d given that the hypothesis h was true. This is known as posterior probability.

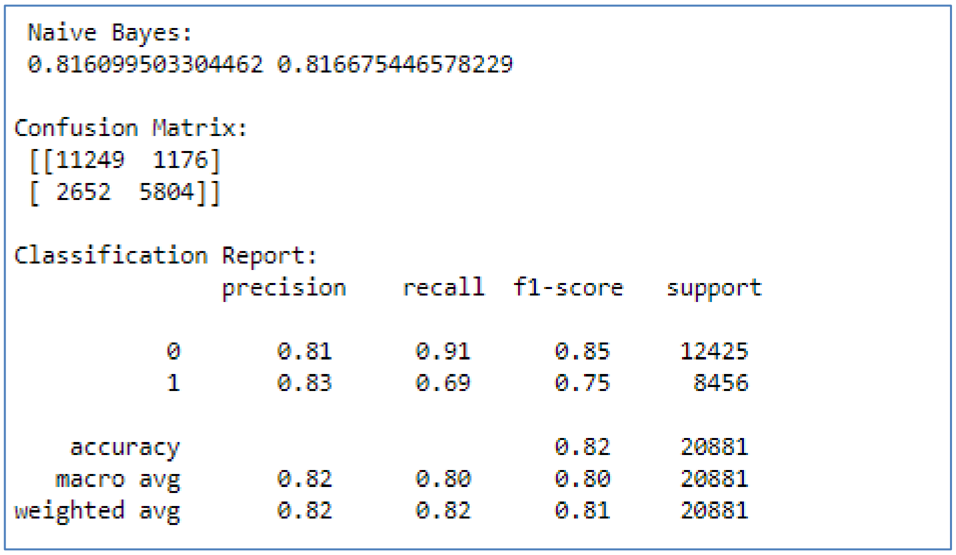
Naive Bayes classifier calculates the probability of an event in the following steps:

- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for each class
- Step 3: Put these value in Bayes Formula and calculate posterior probability.
- Step 4: See which class has a higher probability, given the input belongs to the higher probability class.



Naive Bayes

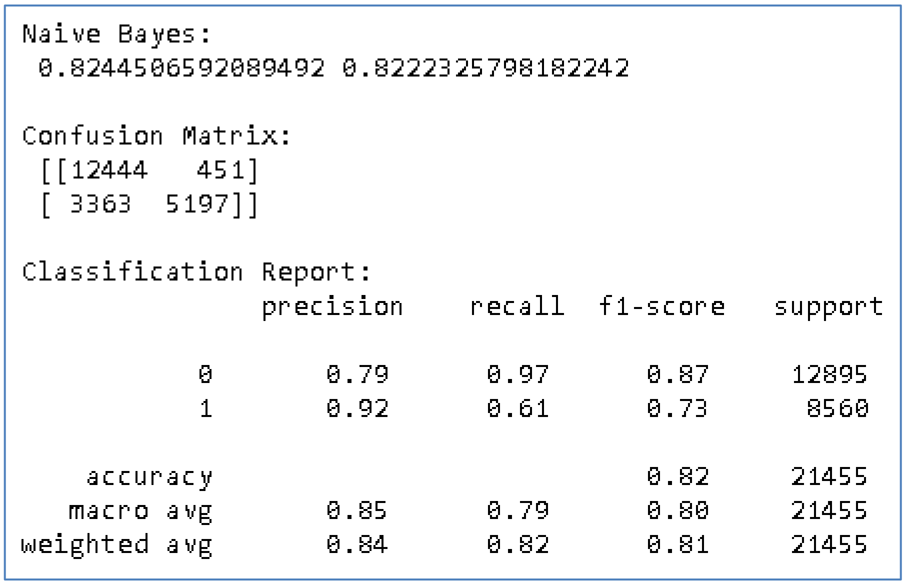
```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB.fit(X_train,y_train)
NB_PRED = NB.predict(X_test) #Confusion matrix & Clssification Report
NB_PRED_PROB = NB.predict_proba(X_test)[:,1] #ROC Curve
print('\n Naive Bayes:\n',NB.score(X_train, y_train),NB.score(X_test, y_test))
print('\nConfusion Matrix: \n',confusion_matrix(y_test,NB_PRED))
print('\nClassification Report: \n',classification_report(y_test,NB_PRED))
```



```
 Naive Bayes:
 0.816099503304462 0.816675446578229

Confusion Matrix:
 [[11249  1176]
 [ 2652  5804]]

Classification Report:
               precision    recall  f1-score   support

           0       0.81      0.91      0.85     12425
           1       0.83      0.69      0.75      8456

    accuracy                           0.82     20881
   macro avg       0.82      0.80      0.80     20881
weighted avg       0.82      0.82      0.81     20881
```

Hyper Parameter Tuning:

- var_smoothing Range between (1e-5 to 1e-6)



```
Naive Bayes:
 0.8244506592089492 0.8222325798182242

Confusion Matrix:
 [[12444   451]
 [ 3363  5197]]

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.97      0.87     12895
           1       0.92      0.61      0.73      8560

    accuracy                           0.82     21455
   macro avg       0.85      0.79      0.80     21455
weighted avg       0.84      0.82      0.81     21455
```

Advantages:

- When assumption of independent predictors holds true, a Naive Bayes classifier performs better as compared to other models.
- Naive Bayes requires a small amount of training data to estimate the test data. So, the training period is less.
- Naive Bayes is also easy to implement.

Decision Tree:

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

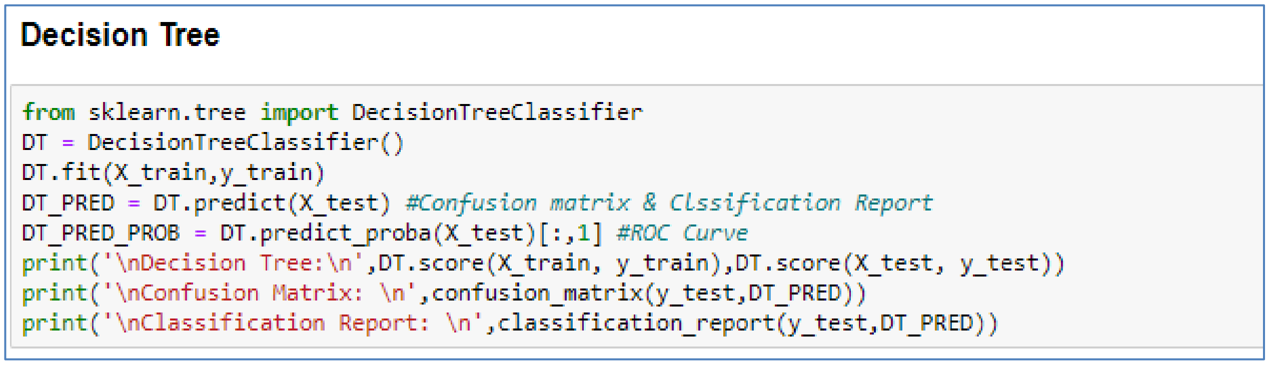
In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees.

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

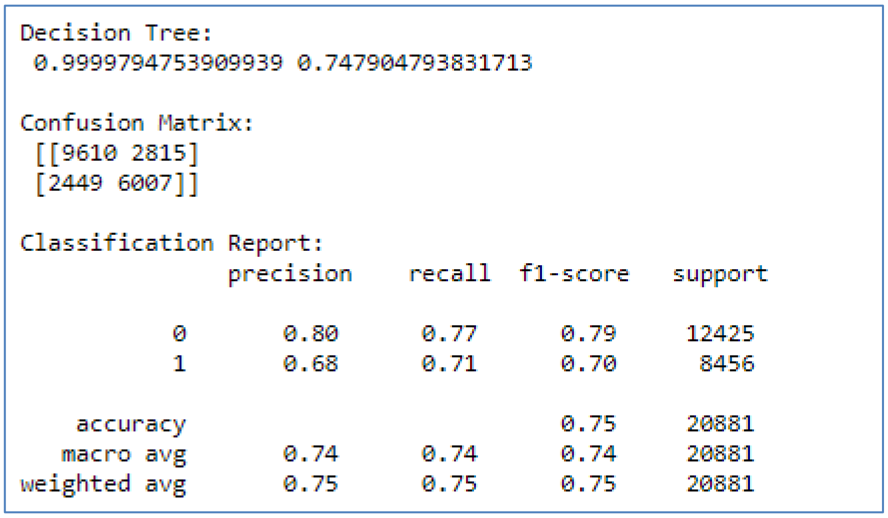
Steps:

1. It begins with the original set S as the root node.
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates Entropy(H) and Information Gain(IG) of this attribute.
3. It then selects the attribute which has the smallest Entropy or Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.
5. The algorithm continues to recur on each subset, considering only attributes never selected before.



Decision Tree

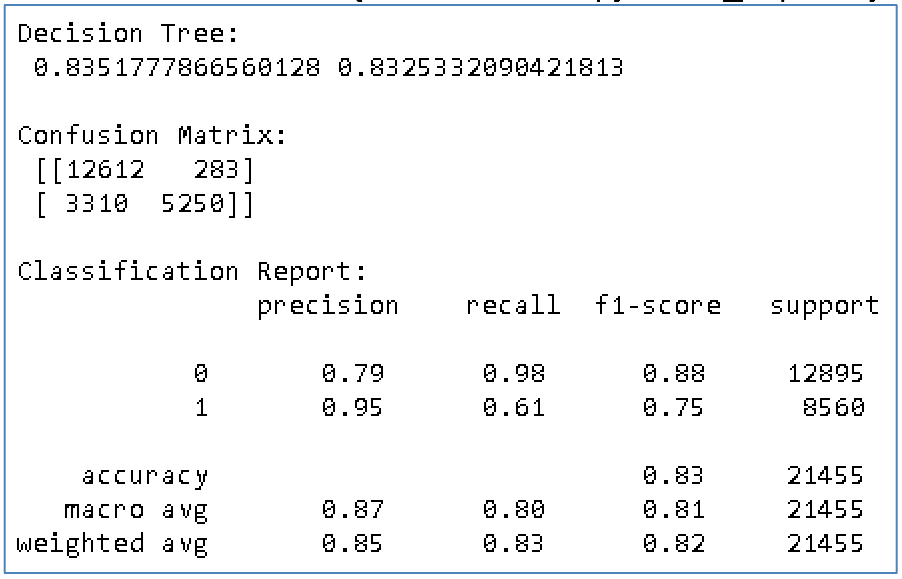
```python
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(X_train,y_train)
DT_PRED = DT.predict(X_test) #Confusion matrix & Clssification Report
DT_PRED_PROB = DT.predict_proba(X_test)[:,1] #ROC Curve
print('\nDecision Tree:\n',DT.score(X_train, y_train),DT.score(X_test, y_test))
print('\nConfusion Matrix: \n',confusion_matrix(y_test,DT_PRED))
print('\nClassification Report: \n',classification_report(y_test,DT_PRED))
```



```
Decision Tree:
 0.9999794753909939 0.747904793831713

Confusion Matrix:
 [[9610 2815]
 [2449 6007]]

Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.77      0.79     12425
           1       0.68      0.71      0.70      8456

    accuracy                           0.75     20881
   macro avg       0.74      0.74      0.74     20881
weighted avg       0.75      0.75      0.75     20881
```

Hyper Parameter Tuning:

Best parameters for the model are: {'criterion': 'entropy', 'max_depth': 4}



```
Decision Tree:
 0.8351777866560128 0.8325332090421813

Confusion Matrix:
 [[12612   283]
 [ 3310  5250]]

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.98      0.88     12895
           1       0.95      0.61      0.75      8560

    accuracy                           0.83     21455
   macro avg       0.87      0.80      0.81     21455
weighted avg       0.85      0.83      0.82     21455
```

<u>Advantages</u>:

● Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
● A decision tree does not require normalization of data.
● A decision tree does not require scaling of data as well.
● Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
● A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

<u>Random Forest Classifier:</u>

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. One big advantage of random forest is that it can be used for both classification and regression problems.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

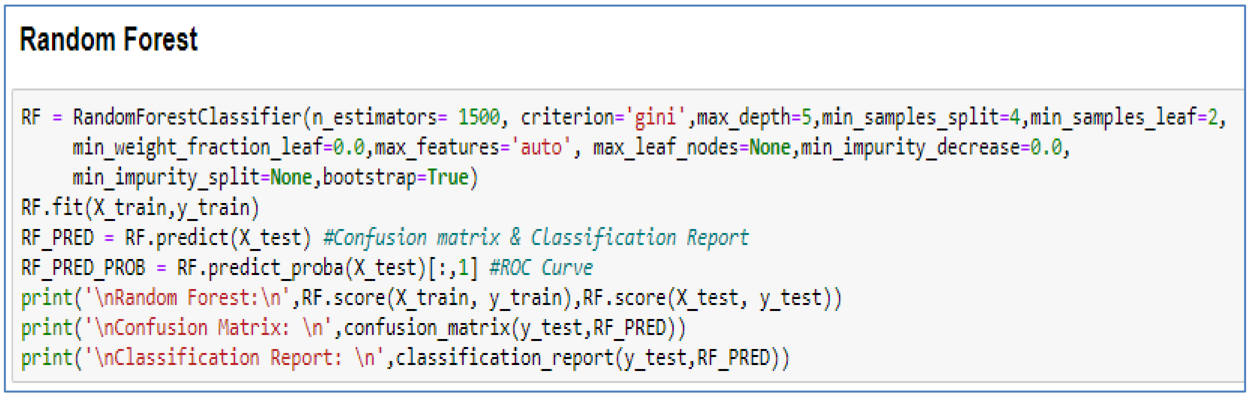
Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

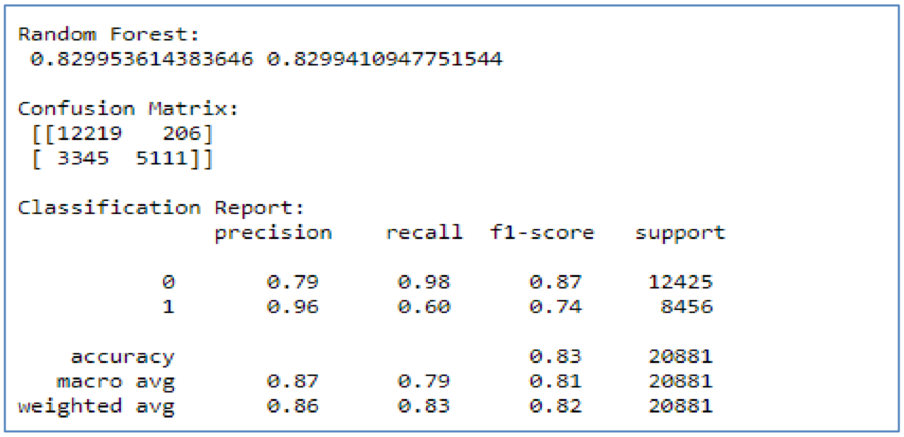
The Working process can be explained in the below steps and diagram:

- Select random K data points from the training set.
- Build the decision trees associated with the selected data points (Subsets).
- Choose the number N for decision trees that you want to build.
- Repeat Step 1 & 2.
- For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



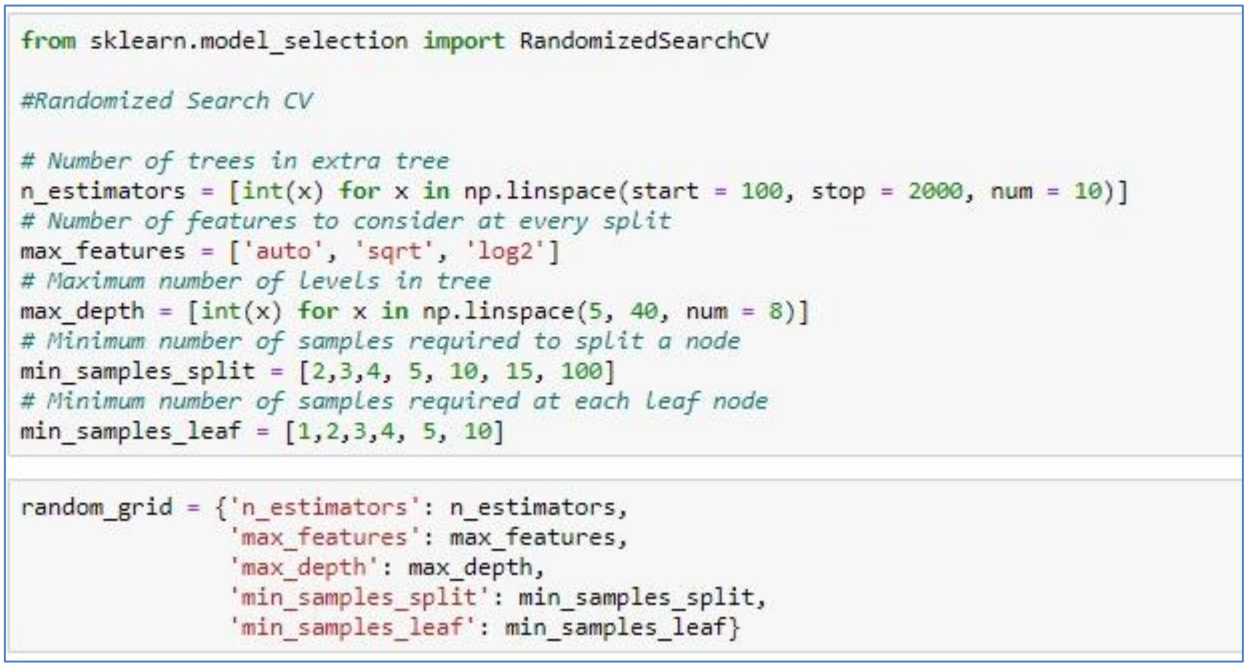
Random Forest

```
RF = RandomForestClassifier(n_estimators= 1500, criterion='gini',max_depth=5,min_samples_split=4,min_samples_leaf=2,
     min_weight_fraction_leaf=0.0,max_features='auto', max_leaf_nodes=None,min_impurity_decrease=0.0,
     min_impurity_split=None,bootstrap=True)
RF.fit(X_train,y_train)
RF_PRED = RF.predict(X_test) #Confusion matrix & Classification Report
RF_PRED_PROB = RF.predict_proba(X_test)[:,1] #ROC Curve
print('\nRandom Forest:\n',RF.score(X_train, y_train),RF.score(X_test, y_test))
print('\nConfusion Matrix: \n',confusion_matrix(y_test,RF_PRED))
print('\nClassification Report: \n',classification_report(y_test,RF_PRED))
```



```
Random Forest:
 0.829953614383646 0.8299410947751544

Confusion Matrix:
 [[12219   206]
 [ 3345  5111]]

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.98      0.87     12425
           1       0.96      0.60      0.74      8456

    accuracy                           0.83     20881
   macro avg       0.87      0.79      0.81     20881
weighted avg       0.86      0.83      0.82     20881
```

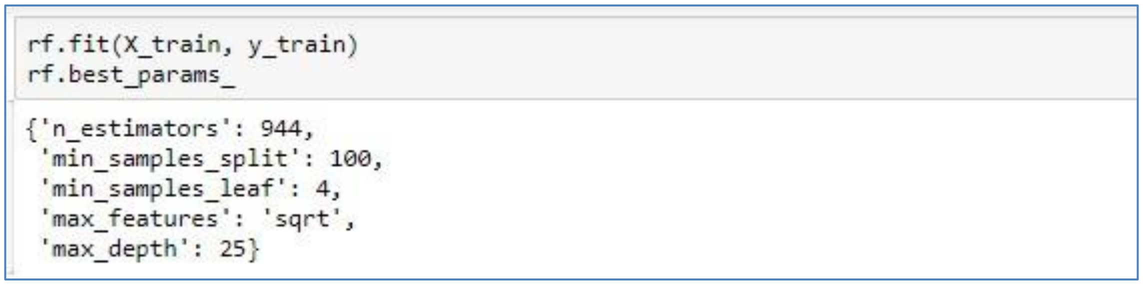
Hyper Parameter Tuning:



```
from sklearn.model_selection import RandomizedSearchCV

#Randomized Search CV

# Number of trees in extra tree
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt', 'log2']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 40, num = 8)]
# Minimum number of samples required to split a node
min_samples_split = [2,3,4, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1,2,3,4, 5, 10]
```

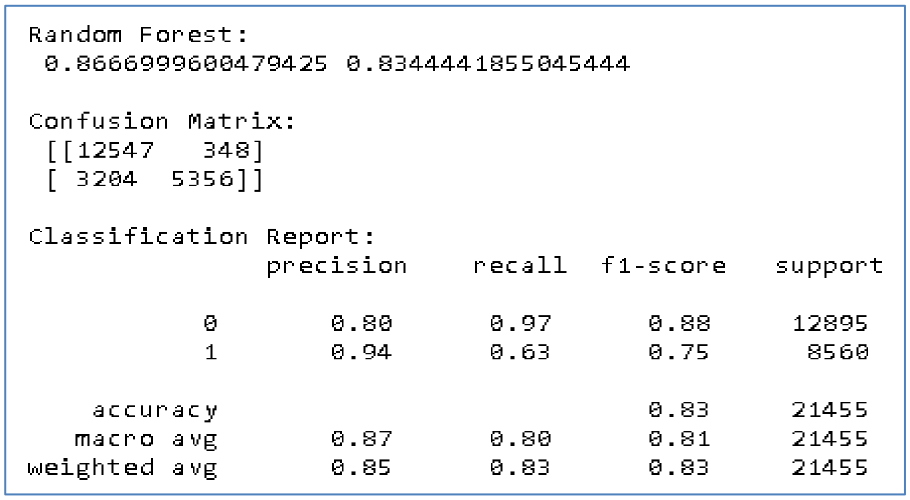
```
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

Best Parameters:



```
rf.fit(X_train, y_train)
rf.best_params_
```

```
{'n_estimators': 944,
 'min_samples_split': 100,
 'min_samples_leaf': 4,
 'max_features': 'sqrt',
 'max_depth': 25}
```



```
Random Forest:
 0.8666999600479425 0.8344441855045444

Confusion Matrix:
 [[12547   348]
 [ 3204  5356]]

Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.97      0.88     12895
           1       0.94      0.63      0.75      8560

    accuracy                           0.83     21455
   macro avg       0.87      0.80      0.81     21455
weighted avg       0.85      0.83      0.83     21455
```

Advantages:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Extreme Gradient Boosting:

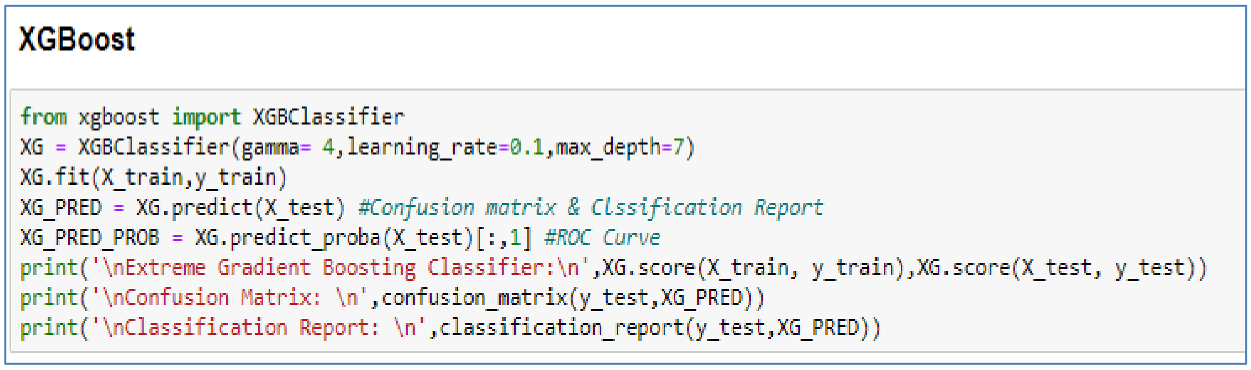
XGBoost is an ensemble learning method. XGBoost working is similar to the Gradient Boosting. XGBoost is the advanced implementation of gradient boosting algorithm along with some regularization factors, which improves model generalization. Here, we have applied XGBoost to solve our low accuracy problem. Hence, XGBoost helped us to get the accuracy what we needed. Below is the screenshot of implementation and the accuracy for that.

Gradient Boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Just like any other machine learning models in Exploratory, you can build, predict, and evaluate XGBoost models.

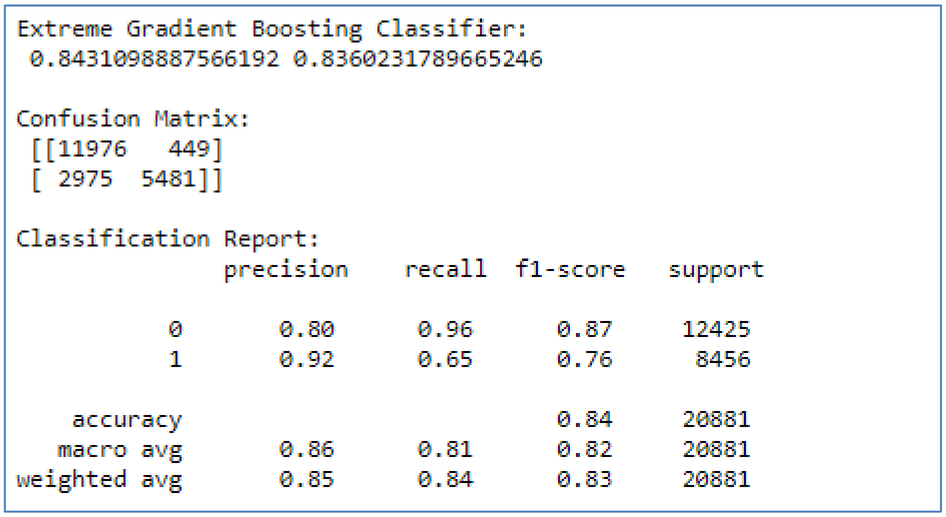
Steps Involved in building the XGBoost Model:

- Prepare Data
- Build a Model with XGBoost
- Predict and Visualize the Predicted Values
- Evaluate with the Prediction Performance metrics and ROC curve
- Compare the Prediction Performance against Model



XGBoost

```
from xgboost import XGBClassifier
XG = XGBClassifier(gamma= 4,learning_rate=0.1,max_depth=7)
XG.fit(X_train,y_train)
XG_PRED = XG.predict(X_test) #Confusion matrix & Clssification Report
XG_PRED_PROB = XG.predict_proba(X_test)[:,1] #ROC Curve
print('\nExtreme Gradient Boosting Classifier:\n',XG.score(X_train, y_train),XG.score(X_test, y_test))
print('\nConfusion Matrix: \n',confusion_matrix(y_test,XG_PRED))
print('\nClassification Report: \n',classification_report(y_test,XG_PRED))
```



```
Extreme Gradient Boosting Classifier:
 0.8431098887566192 0.8360231789665246

Confusion Matrix:
 [[11976   449]
 [ 2975  5481]]

Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.96      0.87     12425
           1       0.92      0.65      0.76      8456

    accuracy                           0.84     20881
   macro avg       0.86      0.81      0.82     20881
weighted avg       0.85      0.84      0.83     20881
```

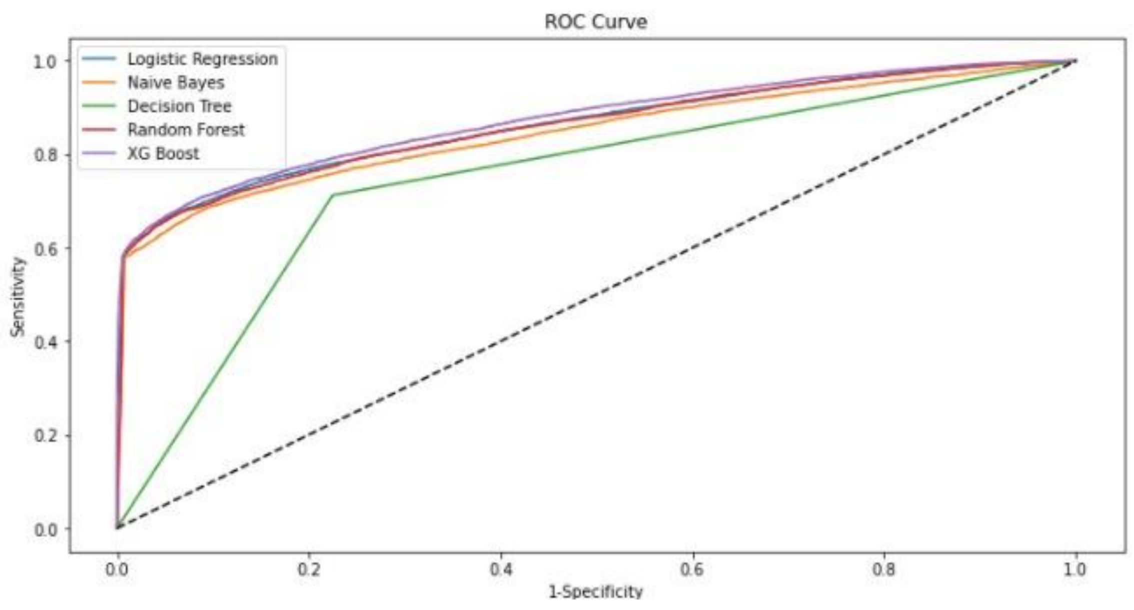
<ins>**Advantages**</ins>**:**

- It is one of the most powerful algorithms with high speed and performance.
- It can harness all the processing power of modern multicore computers.
- It is feasible to train on large datasets.
- Consistently outperform all single algorithm methods.

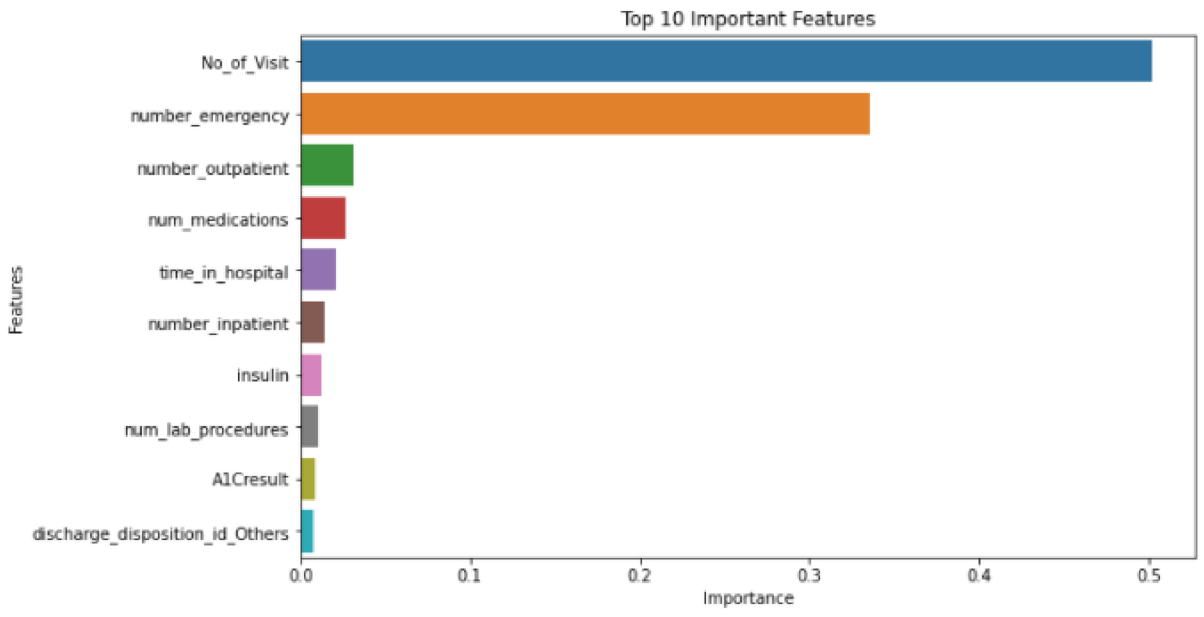
Model Evaluation Metrics:

Model	Accuracy	Precision	Recall	F1-score	Roc-Auc Score
Logistic Regression	0.87	0.95	0.61	0.74	0.84
Naïve Bayes	0.85	0.92	0.61	0.73	0.86
Decision Tree Classifier	0.87	0.95	0.61	0.75	0.74
Random Forest Classifier	0.87	0.94	0.63	0.75	0.87
XG Boost Classifier	0.86	0.92	0.76	0.76	0.88

Model Performance Metrics:



Feature Importance:



- Feature 'No of Visit' created by us based on the unique patient visit, is having Maximum Importance in our Model.
- Followed by Number of Emergency and so on.
- We could observe there are Maximum Numerical Features, which has high correlation with respet to the Target Variable.
- As we saw dominating trends in the Medications with 98% representing Class 0 & 2 or less percent representing Class1.
- Medicines are not a Significant Contributer to our model.
- We observed Categorical Medical terms like Insulin & A1C Result being in top 10 Important Features.
- Both This terms are related to Diabetes, Thus we can say, People readmitted in the hospital has more probability of having Diabetes Disease.

Conclusion:

● Hospital readmission of patients with diabetes is an important health care quality measure and driver of costs.

● Major risk factors for readmission include lower socioeconomic status, racial/ethnic minority, greater burden of comorbidities, public insurance, emergent or urgent admission, and a history of recent prior hospitalization.

● Certain hospitalized patients with diabetes may be at higher risk of readmission than those without diabetes.

● Multiple health system and patient-related barriers to reducing readmission rates exist.

● A mix of expert opinion and a handful of mostly small studies provide a number of potential strategies for reducing readmission risk, including inpatient education, specialty care, better discharge instructions, coordination of care, and post-discharge support.

● The diabetes-specific strategies such as diabetes education, intensifying therapy, and outpatient diabetes care tend to be more effective in poorly controlled patients and tend to reduce.

References:

- 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.

Link: https://www.nejm.org/doi/full/10.1056/NEJMsa1513024#article_references

● World Health Organization - Diabetes

Link: https://www.nejm.org/doi/full/10.1056/NEJMsa1513024#article_references

● Readmissions, Observation, and the Hospital Readmissions Reduction Program

Link: https://www.nejm.org/doi/full/10.1056/NEJMsa1513024#article_references

Notes For Project Team

Sample Reference for Datasets (to be filled by team and mentor)

Original owner of data	
Data set information	
Any past relevant articles using the dataset	
Reference	
Link to web page	

****