A

Capstone Project Report (Final)

On

"DIABETIC PATIENTS' READMISSION PREDICTION"

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. Srikar Muppidi

Submitted by-

Ann Kurillose

Jayesh Gupta

Kartikay Raniwala

Rahul Sharma

Rimjim Razdan

Saurabh Tayal

Candidate's Declaration

We hereby declare that the work, which is being presented in the **Capstone Project Report (Final)**, entitled **Diabetic Patients' Readmission Prediction** in partial fulfillment for the award of degree of **PGP** in **Data Science and Engineering** is a record of our own work/investigations carried under the guidance of Mr. Srikar Muppidi**.**

We have not submitted the matter presented in this Capstone Project Report anywhere for the award of any other Degree.

Group 1

(1) Ann Kurillose

(2) Jayesh Gupta

(3) Kartikay Raniwala

(4) Rahul Sharma

(5) Rimjim Razdan

(6) Saurabh Tayal

LIST OF FIGURES

ABSTRACT

Title: Diabetes Patients' Readmission Prediction

Project Description:

Hospital readmission is an indicator of the quality of care and is a driver for the increasing cost of healthcare. Like other chronic diseases, Diabetes is associated with a higher risk of hospital readmission. In this research, we evaluate several machine learning approaches to predict the probability of hospital re-admissions for diabetic patients. The data set used for this study contains more than 100,000 diabetic patient data and 55 variables including length of stay, insulin, and in-patient visits from hospitals in the United States. We leverage several pre-processing techniques and investigate the performance of the various models. The significant variables contributing to the analysis are the number of in-patients, length of stay, number of medications, number of diagnoses, and age. The results demonstrate the viability of the techniques in providing a better understanding of factors influencing hospital re-admission.

Tools & Technologies used:

Programming Language: Python

IDE: Jupyter Notebook, Google Colaboratory

Visualization: Tableau, Python (Matplotlib and Seaborn)

Models: Logistic Regression, Decision Tree, Random Forest, LightGBM, XGBoost, SVM

Libraries: Numpy, Pandas, Seaborn, Matplotlib, Scikit-learn, Lightgbm, Xgboost, Imblearn, Mlxtend, Time, Warnings, Skater, Shap

Tables of Contents

CHAPTER 1

Introduction to the project

1.1 Background

Diabetes Mellitus (DM) is a chronic disease where the blood has high sugar level. It can occur when the pancreas does not produce enough insulin, or when the body cannot effectively use the insulin it produces (WHO). Diabetes is a progressive disease that can lead to a significant number of health complications and profoundly reduce the quality of life. While many diabetic patients manage the health complication with diet and exercise, some require medications to control blood glucose level. As published by a research article named "The relationship between diabetes mellitus and 30-day readmission rates", it is estimated that 9.3% of the population in the United States have diabetes mellitus (DM), 28% of which are undiagnosed. In recent years, government agencies and healthcare systems have increasingly focused on 30-day readmission rates to determine the complexity of their patient populations and to improve quality. Thirty-day readmission rates for hospitalized patients with DM are reported to be between 14.4 and 22.7%, much higher than the rate for all hospitalized patients (8.5–13.5%).

1.2 Problem Statement

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.

1.3 Impact on business

Hospital readmission is an important contributor to total medical expenditures and is an emerging indicator of quality of care. Diabetes, similar to other chronic medical conditions, is associated with increased risk of hospital readmission. As mentioned in the article "Correction to: Hospital Readmission of Patients with Diabetes", hospital readmission is a high-priority health care quality measure and target for cost reduction, particularly within 30 days of discharge. The burden of diabetes among hospitalized patients 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. Our aim is to provide some insights into the risk factors for readmission and also to identify the medicines that are the most effective in treating diabetes.

CHAPTER 2

Dataset and Domain

2.1 Dataset

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.

Data Source: **UCI Dataset Link**

2.2. Data Dictionary

Column Name	Datatype	Description	Information	Null Values
encounter_id	int64	ID assigned each time a patient arrives at the hospital.	101766 unique values	0%
patient_nbr	int64	Unique ID for every patient.	71518 values	0%
race	object	Race of the patients.	5 categories	2.23%
gender	object	Gender of the patients.	2 categories	0.0029%
age	object	Grouped in 10-year intervals: [0, 10), [10, 20), ..., [90, 100)	10 Bins of size 10	0%
weight	object	Weight in pounds. (Bins)	9 Bins of size 25	96.85%
admission_type_id	int64	Integer identifier corresponding to distinct values, for example, emergency, urgent, elective etc.	8 categories	0%
discharge_disposition_id	int64	Integer identifier corresponding to distinct values, for example, discharged to home etc.	26 categories	0%
admission_source_id	int64	Integer identifier corresponding to distinct values, for example, physician referral, emergency room etc.	17 categories	0%
time_in_hospital	int64	number of days between admission and discharge	1 – 14 Days	0%

payer_code	object	Integer identifier corresponding to distinct values, for example, Blue Cross\Blue Shield, Medicare etc.	23 categories	40%
medical_specialty	object	Integer identifier of a specialty of the admitting physician	73 categories	49%
num_lab_procedures	int64	Number of lab tests performed during the encounter	1 – 132 lab procedures	0%
num_procedures	int64	Number of procedures (other than lab tests) performed during the encounter	0 – 6 procedures	0%
num_medications	int64	Number of distinct generic names administered during the encounter	1 – 81 medications	0%
number_outpatient	int64	Number of outpatient visits of the patient in the year preceding the encounter	0 – 42 times	0%
number_emergency	int64	Number of emergency visits of the patient in the year preceding the encounter	0 – 76 times	0%
number_inpatient	int64	Number of inpatient visits of the patient in the year preceding the encounter	0 – 21 times	0%
diag_1	object	The primary diagnosis (coded as first three digits of ICD9)	717 unique values	0.2%
diag_2	object	Secondary diagnosis (coded as first three digits of ICD9)	749 unique values	0.35%
diag_3	object	Additional secondary diagnosis (coded as first three digits of ICD9)	790 unique values	1.3%
number_diagnoses	int64	Number of diagnoses entered to the system	1 – 16 diagnoses	0%
max_glu_serum	object	The Glucose level in a patient given in (mg/dL).	4 categories	0%
A1Cresult	object	The A1C test result of a patient in percentage.	4 categories	0%
metformin	object		4 categories	0%
repaglinide	object		4 categories	0%
nateglinide	object		4 categories	0%
chlorpropamide	object		4 categories	0%
glimepiride	object		4 categories	0%
acetohexamide	object		2 categories	0%
glipizide	object		4 categories	0%

glyburide	object	(Types of medicines)	4 categories	0%
tolbutamide	object		2 categories	0%
pioglitazone	object		4 categories	0%
Rosiglitazone	object		4 categories	0%
acarbose	object		4 categories	0%
Miglitol	object		4 categories	0%
troglitazone	object		2 categories	0%
Tolazamide	object		3 categories	0%
examide	object		1 category	0%
citoglipton	object		1 category	0%
insulin	object		4 categories	0%
glyburide-metformin	object		4 categories	0%
glipizide-metformin	object		2 categories	0%
glimepiride-pioglitazone	object		2 categories	0%
metformin-rosiglitazone	object		2 categories	0%
metformin-pioglitazone	object		2 categories	0%
Change	object	Indicates if there was a change in diabetic medications	2 categories	0%
diabetesMed	object	Indicates if there was any diabetic medication prescribed.	2 categories	0%
readmitted	object	Days to inpatient readmission	3 categories	0%

NOTE: Since **encounter id** and **patient number** are unique identifiers, we dropped them before proceeding ahead. We are treating each encounter as an independent one since we don't have any time stamp as to when a person was admitted.

CHAPTER 3

Exploration Data Analysis (EDA)

Exploratory Data Analysis (EDA) is the process of visualizing and analyzing data to extract insights from it. In other words, EDA is the process of summarizing important characteristics of data in order to gain better understanding of the dataset.

3.1 Relationship between variables

3.1.1 Univariate Analysis

The analysis of univariate data (type of data consists of **only one variable**) is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it.

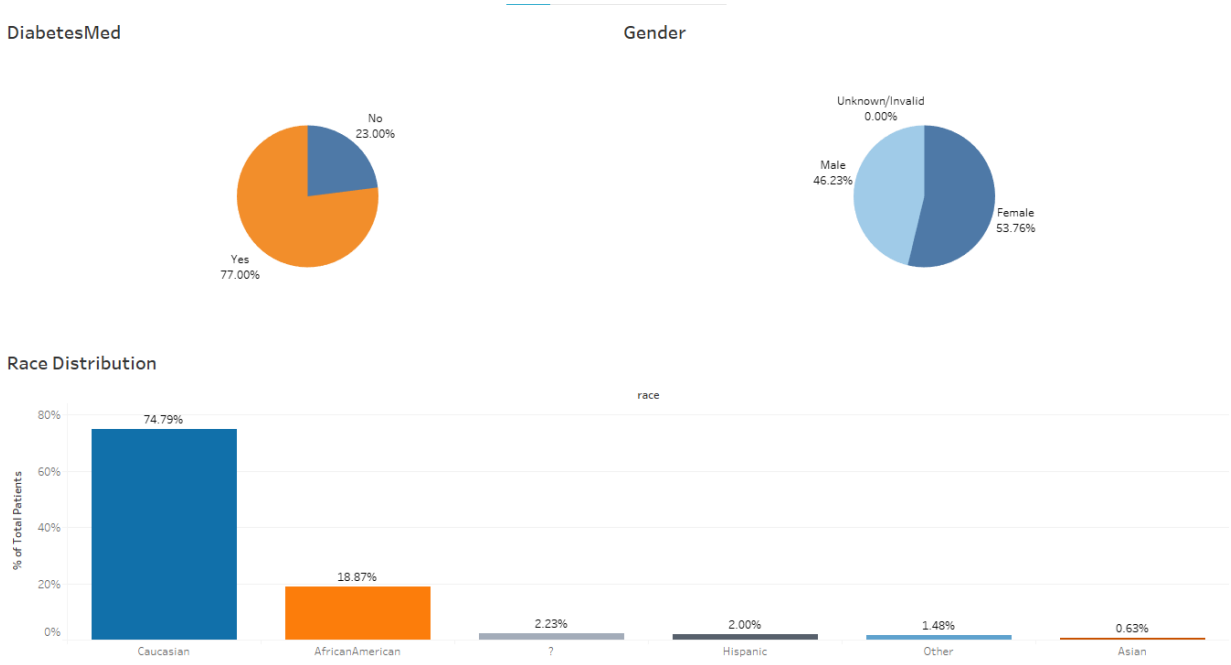

Fig 1: Univariate Plots: race, gender, diabetesMed

- DiabetesMed**:** Diabetes medicines are given to majority (77%) of the patients who are admitted in the hospital.

- Gender: There is almost an equal distribution of male and females that are admitted in the hospital, about 54% females and 46% males. Not a big difference with respect to gender.
- Race: There are 5 different categories in the race feature with the maximum (more than 70%) of them belonging to the Caucasian race.

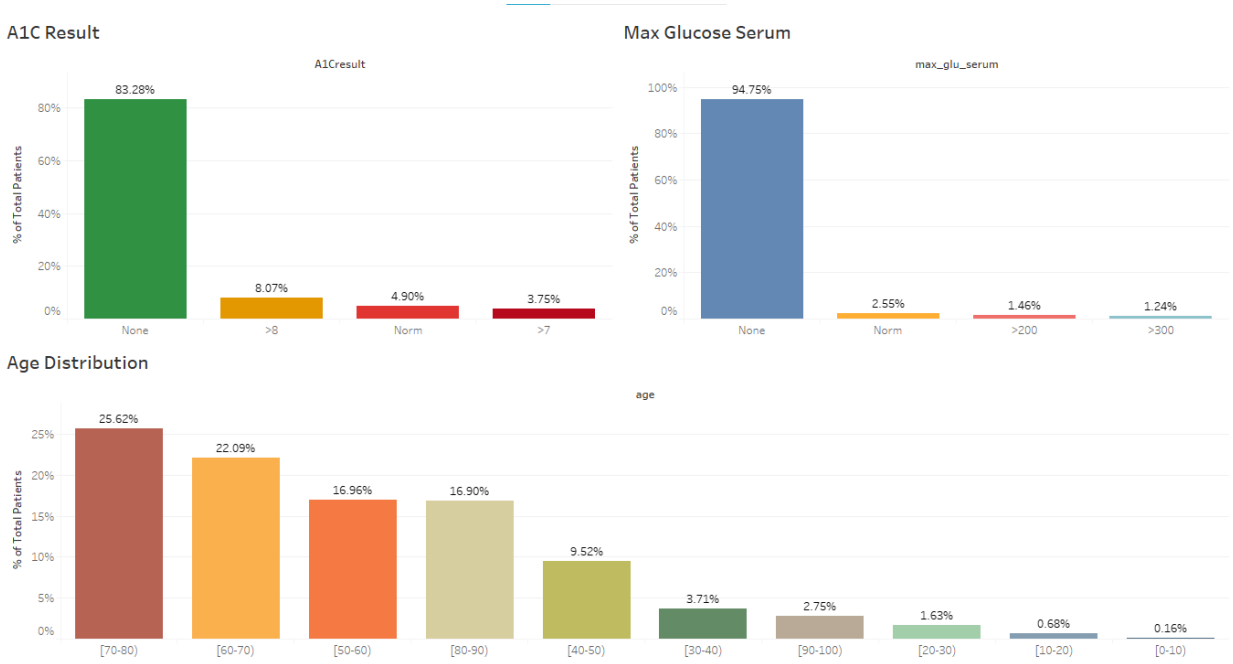

Fig 2: Univariate Plots: age, max_glu_serum, A1Cresult

- A1Cresult: There 4 categories in A1Cresult which indicated the maximum percentage of the patients have not been tested for the A1C test.
- Max_glu_serum: There 4 categories in max_glu_serum which indicated the maximum percentage of the patients have not been tested for the max_glu_serum.
- Age: The maximum patients that have been admitted fall in the range of 50 to 80 yrs.

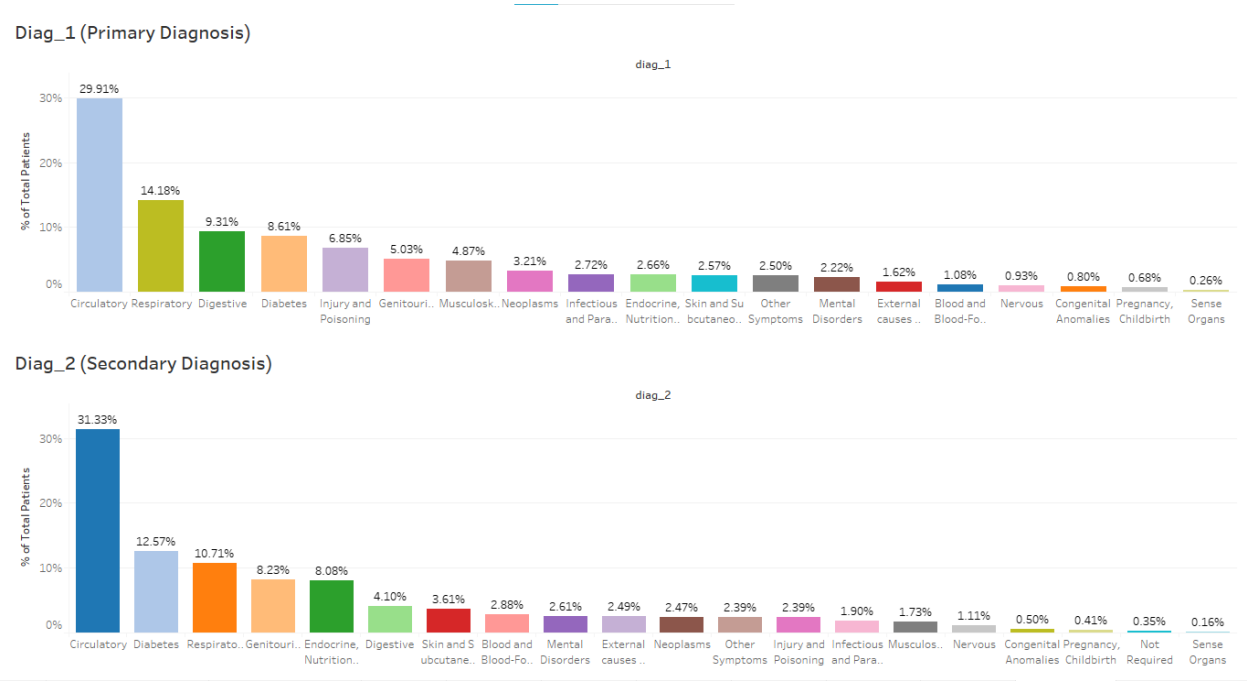

Fig 3: Primary and Secondary Diagnosis

- Circulatory diseases are the ones mostly diagnosed as both primary (~30%) and secondary (~31%) diagnosis indicating that US has a lot of patients with circulatory disease problems.
- 14% of the patients were primarily diagnosed with Respiratory diseases followed by Digestive diseases (~9%).
- About 9% of the patients were primarily diagnosed with Diabetes and ~ 13% of the patients were diagnosed with Diabetes in secondary diagnosis.

3.1.2 Bivariate Analysis

The analysis of bivariate data (type of data involves **two different variables**) deals with causes and relationships and the analysis is done to find out the relationship among the two variables.

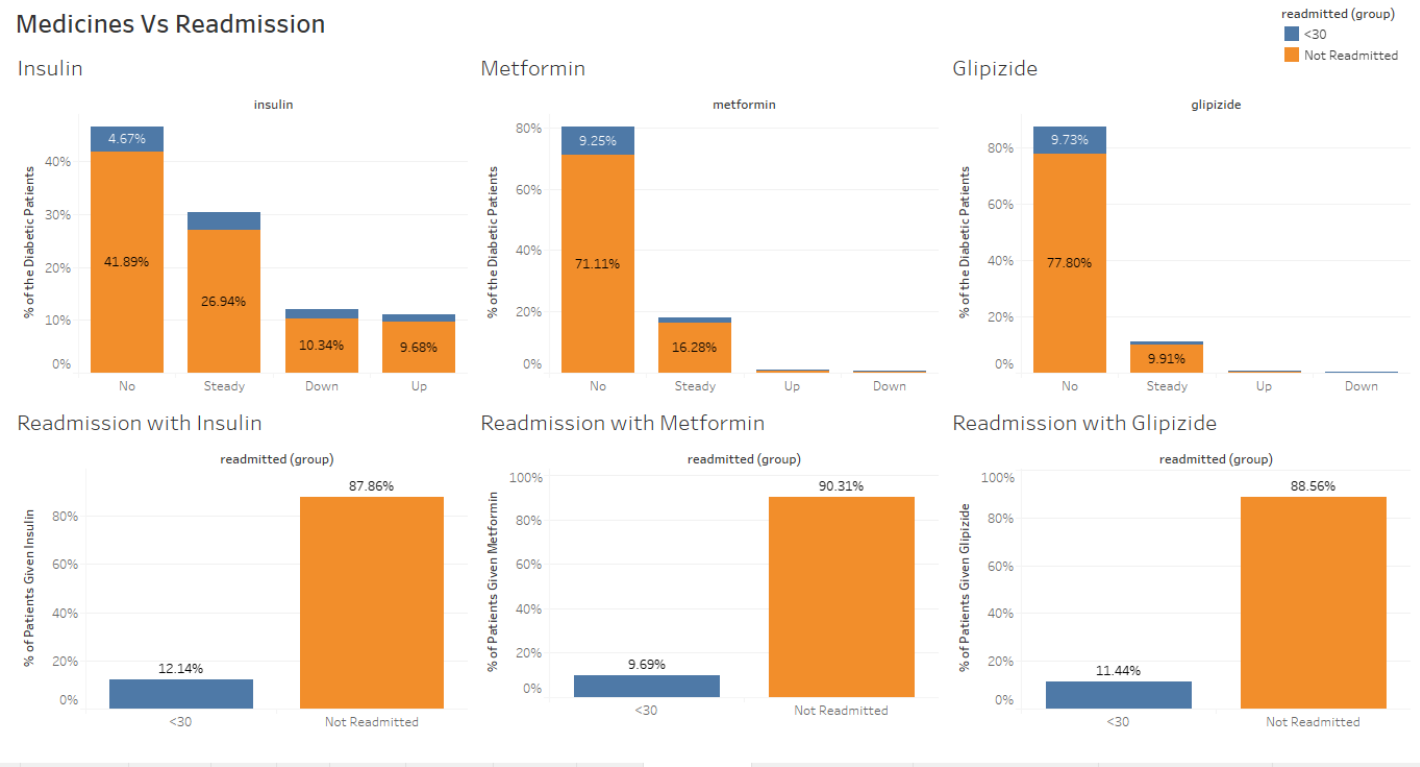

Fig 4: Significant Medicines Vs Readmission

- Above 3 medicines are the most significant as per the chi-square test of independence with respect to the dependent variable 'readmitted' which we plotted with readmitted variable to see the relationship between them. Also, above the medicines are the ones being given to the maximum patients as compared to other medicines.
- Insulin is the most important medicine being given to almost 55% of the patients admitted in the hospital. About 88% of the patients who were given insulin were not readmitted.
- Metformin seems to be 2nd most important medicine after insulin. It is being given to approx. 20% of the patients. About 90% of the patients who were given Metformin were not readmitted.
- Glipizide is another important medicine being given to about 13% of the patients. About 89% of patients who were given Glipizide were not readmitted.

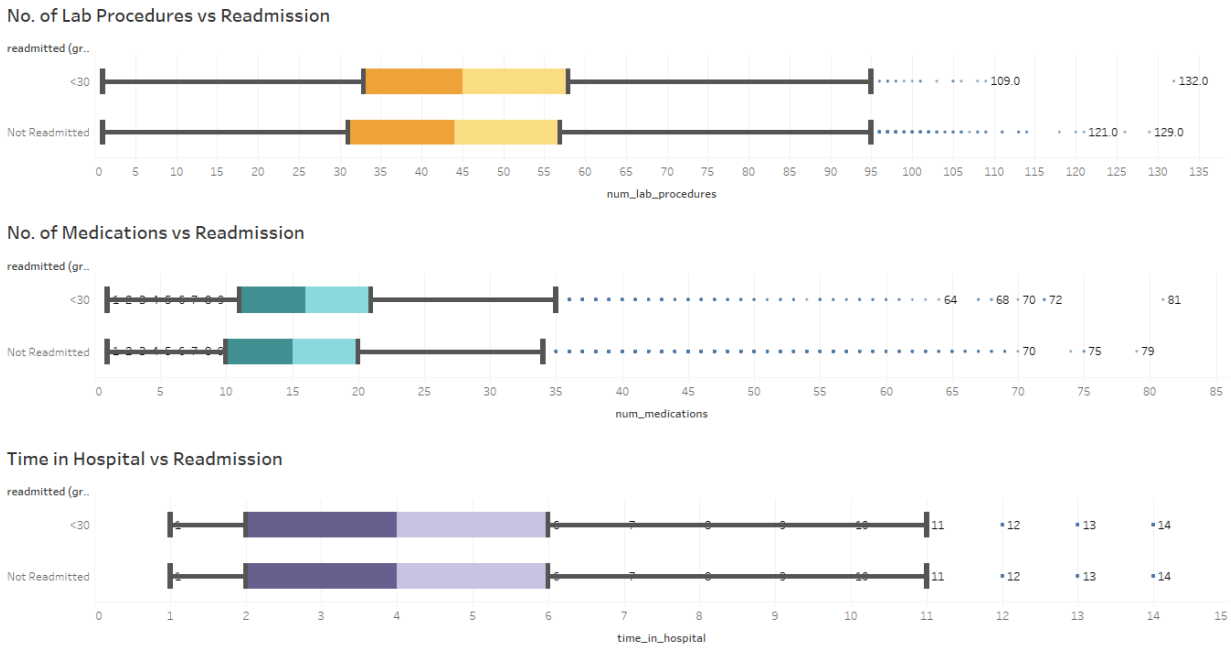

Fig 5: Lab procedures, medications and time vs Readmission

- The average number of lab procedures performed on the patients were slightly higher in case of patients who were readmitted later median being close to 45.
- Average number of medications given were slightly higher in case of readmitted cases, median being close to 15.
- Most of the patients for both groups stayed for 2 – 6 days in the hospital, median being 4 days.

3.2 Check for various criteria

3.2.1 Multi-collinearity

Multicollinearity generally occurs when there are high correlations between two or more predictor variables.
We can identify which variables are affected by multicollinearity and the strength of the correlation. There is a very simple test to assess multicollinearity in a model, the variance inflation factor (VIF). VIF identifies correlation between independent variables and the strength of that correlation.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['VIF']=[variance_inflation_factor(X[num].values,i) for i in range(X[num].shape[1])]
vif['feature']=X[num].columns
vif=vif.sort_values('VIF',ascending=False)
vif
```

	VIF	feature
5	15.085635	number_diagnoses
0	12.141439	age
4	7.651632	num_medications
2	6.101781	num_lab_procedures
1	4.344960	time_in_hospital
3	1.926622	num_procedures
7	1.451683	number_changes
6	1.320692	preceding_year_visits

Fig 6: Multicollinearity-VIF

Heatmap helps us to visualize the correlation between variables. It takes a rectangular data grid as input and then assigns a color intensity to each data cell based on the data value of the cell. This is a great way to get visual clues about the data.

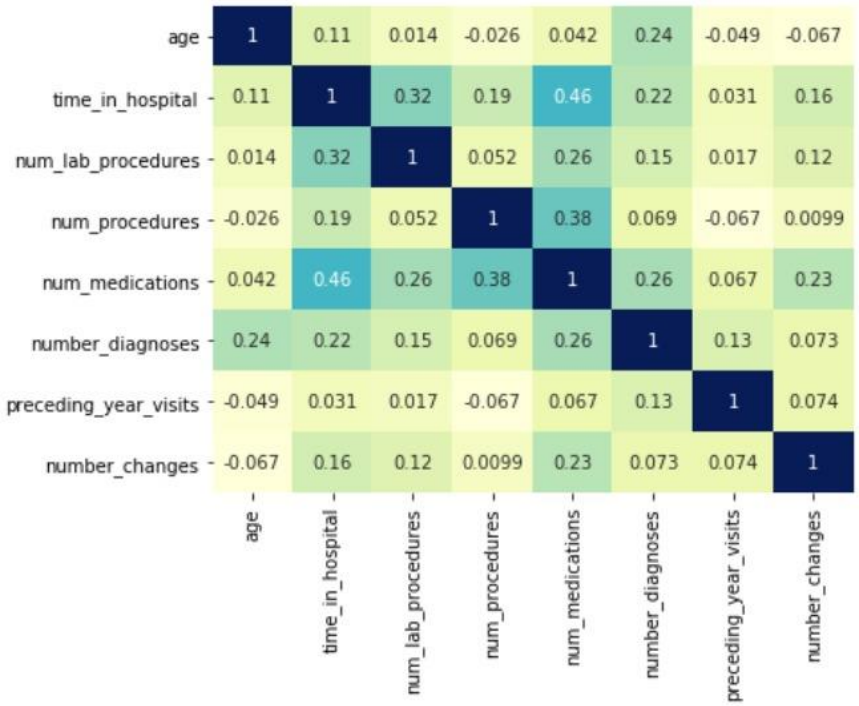

Fig 7: Multicollinearity – Heatmap

- We can see that there is not much correlation between the independent variables. The highest being 0.46 between number of medications and time in hospital.

3.2.2 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.

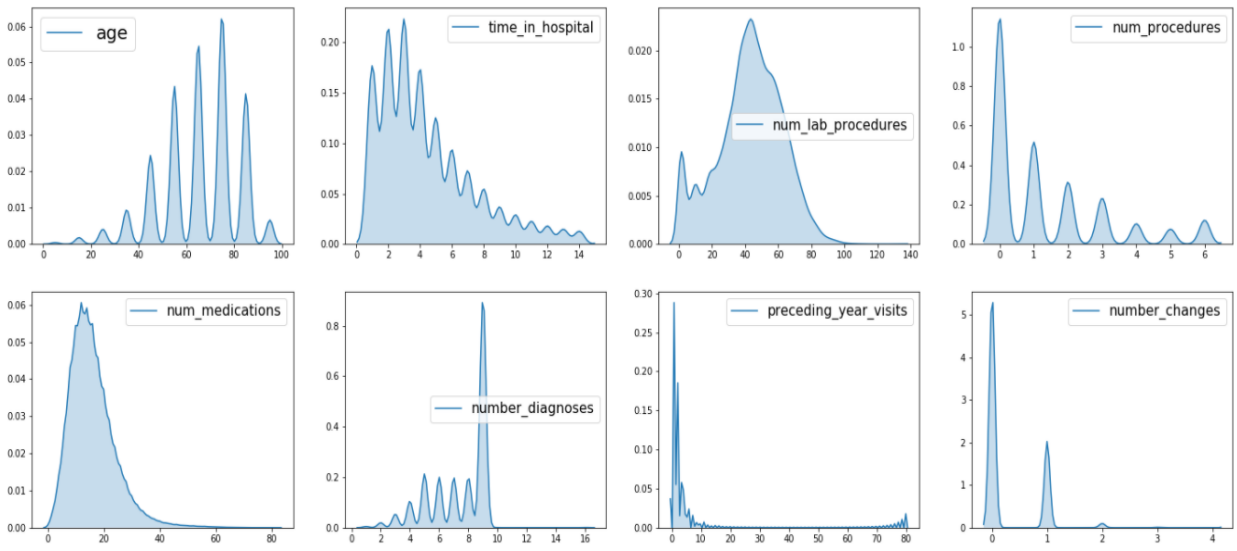

Fig 8: Distribution – KDE

- Age is slightly left skewed and all other variables above are right skewed due to presence of outliers.

3.2.3 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.

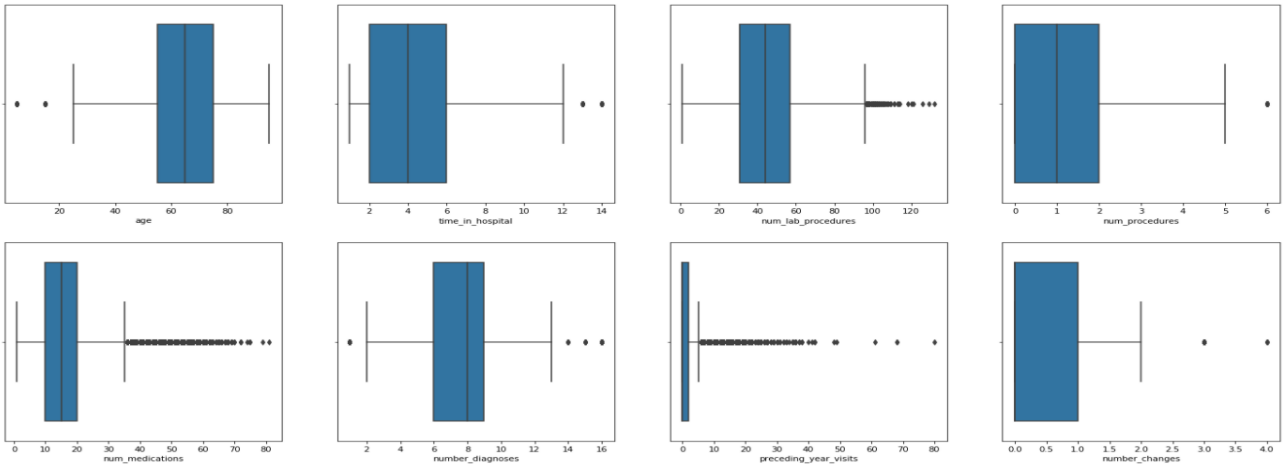

Fig 9: Outliers - Boxplot

- We can see that there are outliers in all of the plots above.
- Age has outliers having low value indicating that its distribution is left skewed.
- Time, number of lab procedures, number of procedures and preceding year visits have outliers with high values making the distribution right skewed.

3.2.4 Class imbalance and its treatment

Class imbalance: This is a scenario where the number of observations belonging to one class is significantly lower than those belonging to the other classes.

A standard machine learning algorithm like Decision Tree and Logistic Regression have a bias towards classes which have number of instances. They tend to only predict the majority class data. The features of the minority class are treated as noise and are often ignored. Thus, there is a high probability of misclassification of the minority class as compared to the majority class.

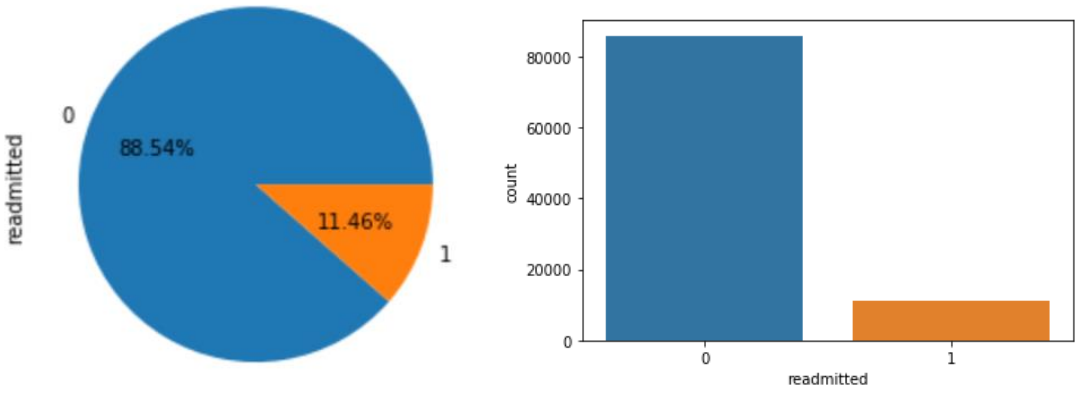

Fig 10: Class Imbalance

Imbalance Ratio ~ 1: 8

Treatment:

- **Oversampling:** Oversampling methods duplicate examples in the minority class or synthesize new examples from the examples in the minority class. Some of the more widely used and implemented oversampling methods include Random Oversampling and Synthetic Minority Oversampling Technique (SMOTE). In some cases, seeking a balanced distribution for a severely imbalanced dataset can cause affected algorithms to overfit the minority class, leading to increased generalization error. The effect can be better performance on the training dataset, but worse performance on the holdout or test dataset. The increase in the number of examples for the minority class, especially if the class skew was severe, can also result in a marked increase in the computational cost when fitting the model, especially considering the model is seeing the same examples in the training dataset again and again.

- **Undersampling:** Undersampling methods delete or select a subset of examples from the majority class. Some of the more widely used and implemented undersampling methods include: Random Undersampling, Condensed Nearest Neighbor Rule (CNN), Near Miss Undersampling, Tomek Links Undersampling, Edited Nearest Neighbors Rule (ENN), One-Sided Selection (OSS), Neighborhood Cleaning Rule (NCR). A limitation of undersampling is that examples from the majority class are deleted that may be useful, important, or perhaps critical to fitting a robust decision boundary. Given that examples are deleted randomly, there is no way to detect or preserve “good” or more information-rich examples from the majority class.

- **Cost Sensitive Algorithms:** Cost-sensitive algorithms are modified versions of machine learning algorithms designed to take the differing costs of misclassification into account when fitting the model on the training dataset. These algorithms can be effective when

used on imbalanced classification, where the cost of misclassification is configured to be inversely proportional to the distribution of examples in the training dataset. There are many cost-sensitive algorithms to choose from, although it might be practical to test a range of cost-sensitive versions of linear, nonlinear, and ensemble algorithms. Some examples of machine learning algorithms that can be configured using cost-sensitive training include: Logistic Regression, Decision Trees, Support Vector Machines, Random Forest, LightGBM, XGBoost.

Note: Keeping in mind the disadvantages of sampling techniques, we have used cost sensitive algorithms to deal with class imbalance.

CHAPTER 4

Pre-Processing Data Analysis

As real-world medical data are often noisy, a particular focus will be led on pre-processing task handling both missing data and inconsistencies but also by reducing the dataset and optimizing it for further model deployment.

4.1 Missing values

The first step in cleaning the data consist of handling missing values. Missing values refers to the absence, voluntary or not, of data in a record. In this data missing values are mainly in the form of question marks (‘? ‘).

- We will drop columns **‘Weight’**, **‘Payer code’** and **‘Medical specialty’** columns as they have more than 40% of missing values.
- In the **‘Gender’** column, there are only three missing values hence we drop them.
- Missing values of the column **‘diag_1’** is replace with ‘Missing’ and Missing values of the column ‘diag_2’ and ‘diag_3’ are replaced with ‘Not required’ if diag_1 and diag_2 value are given respectively. After that we were left with only 1 observation with missing diagnoses in all the 3, hence that observation was dropped.

```
for i in range(len(df['diag_2'])):
    if df.loc[i, 'diag_2'] == 'Missing':
        if df.loc[i, 'diag_1'] != 'Missing':
            df.loc[i, 'diag_2'] = 'Not Required'
```

```
for i in range(len(df['diag_3'])):
    if df.loc[i, 'diag_3'] == 'Missing':
        if df.loc[i, 'diag_2'] == 'Not Required' or df.loc[i, 'diag_2'] != 'Missing':
            df.loc[i, 'diag_3'] = 'Not Required'
```

Fig 11: Filling Missing Values in diag2 and diag_3

- In the ‘**Race’** column, we drop the missing values since we have only 2.23% of missing values in that particular column. Given the large dataset, we can afford to drop them. Moreover, dropping them gave better performance when compared with baseline model.

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

```
Caucasian          76099
AfricanAmerican    19210
?                   2273
Hispanic            2037
Other               1506
Asian                641
Name: race, dtype: int64
```

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

```
Caucasian          76099
AfricanAmerican    19210
Hispanic            2037
Other               1506
Asian                641
Name: race, dtype: int64
```

Fig 12: Dropping missing race values

4.2 Combining similar categories within variables

After having cleaned the data from missing values, it is important to optimize the features and, mostly in this case, reduce the number of unique values for categorical variables.

- We can merge categories of 'admission_type_id', 'admission_source_id' and 'discharge_disposition_id' into fewer number of categories as:

Admission Type Id:

Emergency	• Emergency • Urgent • Trauma Center
Not Available	• Not Available • Null • Not Mapped
Elective	• Elective
New Born	• New Born

```
df.admission_type_id.value_counts()
```

```
1    52900
3    18507
2    17786
6     5225
5     4727
8      317
7       20
4       10
Name: admission_type_id, dtype: int64
```

```
df.admission_type_id.value_counts()
```

```
Emergency        70706
Elective         18507
Not Available    10269
Name: admission_type_id, dtype: int64
```

Fig 13: Combining categories in admission_type_id

Admission Source Id:

Referral	• Physician Referral

	• Clinic Referral • HMO Referral (Health Maintenance Organization)
Transferred from another heal thcare facility	• Transfer from a hospital • Transfer from a Skilled Nursing Facility • Transfer from another health care facility • Transfer from critical access hospital • Transfer from Another Home Health Agency • Readmission to Same Home Health Agency • Transfer from hospital input/same facility resulting in a separate claim • Transfer from Ambulatory Surgery Centre • Transfer from Hospice
Emergency	• Emergency Room • Court/Law Enforcement
Not Available	• Not Available • Not Available • NULL • Not Mapped • Unknown/Invalid
Delivery	• Normal Delivery • Premature Delivery • Sick Baby • Extramural Birth • Born inside this hospital • Born outside this hospital

Discharge Disposition Id:

Discharged to home	• Discharged to home
Transferred to another medica l facility	• Discharged/transferred to another short-term hospital • Discharged/transferred to SNF (skilled nursing facility) • Discharged/transferred to ICF (intermediate care facility) • Discharged/transferred to another type of inpatient car e institution • Neonate discharged to another hospital for neonatal aft ercare • Discharged/transferred/referred another institution for outpatient services

	• Discharged/transferred to another rehab fac including r ehab units of a hospital. • Discharged/transferred to a long-term care hospital. • Discharged/transferred to a nursing facility certified un der Medicaid but not certified under Medicare. • Discharged/transferred to a federal health care facility. • Discharged/transferred/referred to a psychiatric hospit al of psychiatric distinct part unit of a hospital • Discharged/transferred to a Critical Access Hospital (CAH). • Discharged/transferred to another Type of Health Care Institution not defined Elsewhere.
Left AMA (Against Medical Advice.)	• Left AMA (Against Medical Advice.)
Discharged to home with ho me health service	• Discharged/transferred to home with home health servi ce • Discharged/transferred to home under care of Home I V provider
Still patient/referred to this in stitution	• Admitted as an inpatient to this hospital • Still patient or expected to return for outpatient service s • Discharged/transferred within this institution to Medic are approved swing bed • Discharged/transferred/referred to this institution for o utpatient services
Expired	• Expired • Expired at home. Medicaid only, hospice. • Expired in a medical facility. Medicaid only, hospice. • Expired, place unknown. Medicaid only, hospice.
Not Available	• NULL • Not Mapped • Unknown/Invalid
Hospice	• Hospice / home • Hospice / medical facility

- We will cluster the diagnosis codes, namely “diag_1”, “diag_2” and “diag_3”, from the ICD-9 -CM format into fewer comorbidity features. Indeed, each diagnosis variable comprises more than 700 individual ICD codes which make it unusable for further modeling and interpretation. Hence, the diagnoses codes were collapsed in 19 categories including “Circulatory, Respiratory, Digestive, Diabetes, Injury, Musculoskeletal, Genitourinary, Neoplasms etc.

- Since we have 2 columns regarding admission, type_id and source_id, and both have emergency as a value, we used this information to fill the 'Not Available' values in these columns with respect to the other one. For example, if admission_type_id is Emergency but admission_source_id is Not Available, we can fill Emergency in that place.
- After filling it admission_type_id column was dropped since it was giving the same information as admission_source_id and was less significant statistically.

```
for i in diag_cols:
    for j in range(len(df[i])):
        if str(df.loc[j, i])[:3] == '250':
            df.loc[j, i] = 'Diabetes'
        elif ord(str(df.loc[j, i])[0]) in range(69, 87):
            df.loc[j, i] = 'External causes of injury'
        elif df.loc[j, i] == '?':
            df.loc[j, i] = 'Missing'
        else:
            x = float(df.loc[j, i])
            if x in range(390, 460) or x == 785:
                df.loc[j, i] = 'Circulatory'
            elif x in range(460, 520) or x == 786:
                df.loc[j, i] = 'Respiratory'
            elif x in range(520, 580) or x == 787:
                df.loc[j, i] = 'Digestive'
            elif x in range(800, 1000):
                df.loc[j, i] = 'Injury and Poisoning'
            elif x in range(710, 740):
                df.loc[j, i] = 'Musculoskeletal System and Connective Tissue'
            elif x in range(580, 630) or x == 788:
                df.loc[j, i] = 'Genitourinary'
            elif x in range(140, 230):
                df.loc[j, i] = 'Neoplasms'
            elif x in [780, 781, 784] or x in range(790, 800):
                df.loc[j, i] = 'Other Symptoms'
            elif x in range(240, 280):
                df.loc[j, i] = 'Endocrine, Nutritional, Metabolic, Immunity'
            elif x in range(680, 710) or x == 782:
                df.loc[j, i] = 'Skin and Subcutaneous Tissue'
            elif x in range(1, 140):
                df.loc[j, i] = 'Infectious and Parasitic'
            elif x in range(290, 320):
                df.loc[j, i] = 'Mental Disorders'
            elif x in range(280, 290):
                df.loc[j, i] = 'Blood and Blood-Forming Organs'
            elif x in range(320, 360):
                df.loc[j, i] = 'Nervous'
            elif x in range(630, 680):
                df.loc[j, i] = 'Pregnancy, Childbirth'
            elif x in range(360, 390):
                df.loc[j, i] = 'Sense Organs'
            else:
                df.loc[j, i] = 'Congenital Anomalies'
```

Fig 14: Creating diagnosis categories based on ICD 9 Codes.

4.3 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.

4.4 Encoding

- We converted all the **medicine** columns into numerical ones by encoding them as:
  ‘Not Given’: -2,
  ‘Down’: -1,
  ‘No Change’: 0,
  ‘Up’: 1.
- We label encoded ‘**gender’** column as **Female** : 0, **Male** : 1.
- We label encoded columns ‘**change’** and ‘**diabetesMed’** as follows:

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

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

Fig 15: Encoding Binary Columns

- **Age** was given in classes, so we replaced them with their respective means.
- **Readmission**: The outcome we are looking at is whether the patient gets readmitted to the hospital within 30 days or not. So, with respect to our problem we will convert this into 2 categories i.e. No readmission or readmission within 30 days.

4.5 Creating New Features and Dropping Redundant Ones

There are three new features which have been created:

1. **'preceding_year_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.
2. **'number_changes':** 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_changes'] = np.nan
for i in range(len(df)):
    n = 0
    for j in meds:
        if df.loc[i, j] == -1 or df.loc[i, j] == 1:
            n += 1
    df.loc[i, 'number_changes'] = n
```

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

```
0.0    70599
1.0    25081
2.0     1279
3.0      106
4.0        5
Name: number_changes, dtype: int64
```

Fig 16: Code: Creating 'number_changes'

3. **'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'
df['insulin_treatment'].value_counts()
```

```
insulin_only     29118
other_meds       23259
insulin_combo    22465
no_med           22228
Name: insulin_treatment, dtype: int64
```

Fig 17: Code: Creating 'insulin_treatment'

4.6 Statistical significance of variables and removal based on it.

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)

p_val = []
sig = []
for i in df.columns:
    if i in num_cols:
        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
```

Fig 18: Statistical Tests

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.

	columns	p_value	significance
38	preceding_year_visits	0.000000e+00	Significant
37	readmitted	0.000000e+00	Significant
3	discharge_disposition_id	1.002865e-175	Significant
12	number_diagnoses	1.181113e-62	Significant
5	time_in_hospital	6.893070e-49	Significant
30	insulin	1.450024e-43	Significant
8	num_medications	1.128847e-39	Significant
40	insulin_treatment	1.750135e-39	Significant
39	number_changes	3.627420e-30	Significant
11	diag_3	5.426864e-30	Significant
9	diag_1	7.642047e-28	Significant
10	diag_2	1.229337e-19	Significant
36	diabetesMed	2.949361e-16	Significant
15	metformin	2.759957e-15	Significant
6	num_lab_procedures	3.503248e-14	Significant
2	age	3.275694e-11	Significant
4	admission_source_id	1.937846e-09	Significant
14	A1Cresult	1.150395e-08	Significant
13	max_glu_serum	2.682172e-04	Significant
7	num_procedures	1.530071e-03	Significant
21	glipizide	6.078898e-03	Significant
16	repaglinide	1.001026e-02	Significant
19	glimepiride	4.203875e-02	Significant
24	pioglitazone	4.487769e-02	Significant

	columns	p_value	significance
25	rosiglitazone	7.245006e-02	Insignificant
22	glyburide	1.143482e-01	Insignificant
0	race	1.263105e-01	Insignificant
27	miglitol	1.688638e-01	Insignificant
35	metformin-pioglitazone	2.261856e-01	Insignificant
20	acetohexamide	2.261856e-01	Insignificant
33	glimepiride-pioglitazone	2.261856e-01	Insignificant
26	acarbose	2.389868e-01	Insignificant
1	gender	4.095739e-01	Insignificant
18	chlorpropamide	4.419945e-01	Insignificant
23	tolbutamide	5.785498e-01	Insignificant
17	nateglinide	6.970633e-01	Insignificant
31	glyburide-metformin	7.088183e-01	Insignificant
29	tolazamide	7.391046e-01	Insignificant
28	troglitazone	7.768704e-01	Insignificant
32	glipizide-metformin	9.925195e-01	Insignificant
34	metformin-rosiglitazone	1.000000e+00	Insignificant

Fig 19: Statistical Significance

NOTE: The following 13 medicines have been found statistically insignificant and have been given to less than 1% of the population and hence has been removed from the modelling since it's very unlikely that these medicines will be prescribed to many patients and also they won't add any valuable information to the model.

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

CHAPTER 5

Feature Engineering

5.1 Transformations

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.

To transform data, we perform a mathematical operation on each observation. There are an infinite number of transformations we could use, but it is better to use a transformation that other researchers commonly use, such as the square-root transformation for count data or the log transformation for size data.

We applied square root transformation to five variables whose skew was greater than 1.

	Skew Before	Skew After
time_in_hospital	1.141139	0.476940
num_procedures	1.320396	0.406015
num_medications	1.331329	0.347681
preceding_year_visits	5.327782	1.164042
number_changes	1.425972	1.077153

Fig 20: Transformation

5.2 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. MinMaxScaler normalizes the data to the range of [0, 1] by subtracting the minimum value and then dividing by range. This is done only for numerical variables.

```
mm = MinMaxScaler()
X_train = pd.DataFrame(mm.fit_transform(X_train), columns = X_train.columns)
X_test = pd.DataFrame(mm.transform(X_test), columns = X_test.columns)
X_train.head()
```

Fig 21: Scaling

5.3 Feature Selection

Feature selection is for filtering irrelevant or redundant features from your dataset.

Several techniques were tested to perform feature selection including statistical significance tests, forward selection, backward elimination and feature importance. Based on the collective results of these techniques, below are the set of most important features selected for model building.

SET OF SELECTED FEATURES

➢ num_lab_procedures	➢ diag_1_Circulatory
➢ num_medications	➢ diabetesMed
➢ time_in_hospital	➢ discharge_disposition_id_Discharged to home with home health service
➢ preceding_year_visits	➢ diag_3_Circulatory
➢ age	➢ admission_source_id_Referral
➢ number_diagnoses	➢ glipizide
➢ num_procedures	➢ number_changes
➢ discharge_disposition_id_Transferred to another medical facility	➢ glyburide
➢ insulin_treatment_insulin_only	➢ diag_1_Genitourinary
➢ race_Caucasian	➢ A1Cresult_None
➢ metformin	➢ diag_1_External causes of injury
➢ gender	➢ discharge_disposition_id_Not Available
➢ diag_1_Respiratory	➢ insulin

CHAPTER 6

Model building

In machine learning, there's something called the **"No Free Lunch"** theorem which states that no one algorithm works best for every problem, and it's especially relevant for supervised learning (i.e. predictive modelling).

For example, you can't say that neural networks are always better than decision trees or vice-versa. There are many factors at play, such as the size and structure of your dataset.

As a result, you should **try many different algorithms for your problem**, while using a hold-out "test set" of data to evaluate performance and select the winner.

Model selection is a method for setting a blueprint to analyse data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.

6.1 Split the data to train and test

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.

stratify : If not None, data is split in a stratified fashion, using this as the class labels.

Splitting the data using sklearn.model_selection (train_test_split): 70% as train data, 30% as test data and random state = 0.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0, stratify = y)
```

6.2 Modeling

Our choice of models is governed primarily by our aim to understand the most important factors, along with their relative effects on medication change and readmission. Therefore, we did not implement models that have little or no interpretability (neural networks, support vector machines, nearest neighbours etc). The models that we implemented include the following.

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. Parallel learning supported.

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) Support Vector Machines: A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Strengths: SVM works relatively well when there is a clear margin of separation between classes. SVM is more effective in high dimensional spaces. SVM is effective in cases where the number of dimensions is greater than the number of samples. SVM is relatively memory efficient.

Weaknesses: SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform. As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

CHAPTER 7

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.

7.1 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 behaviour 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.

Evaluation of different models:

COST SENSITIVE ALGORITHMS (Before Tuning)		
Logistic Regression (class_weight = 'balanced')		
Metrics	Train Set Results	Test Set Results
Accuracy Score	62.47%	62.80%
F1 Score	26.71%	26.82%
Precision Score	17.20%	17.31%
Recall Score	59.69%	59.50%
ROC AUC Score	65.21%	65.65%
Training Time	12.88 seconds	
Prediction Time	0.03 seconds	

Decision Tree Classifier (class_weight = 'balanced')		
Metrics	Train Set Results	Test Set Results
Accuracy Score	99.99%	79.79%
F1 Score	99.99%	16.68%
Precision Score	99.98%	15.81%
Recall Score	100%	17.65%
ROC AUC Score	99.99%	52.74%
Training Time	1.37 seconds	
Prediction Time	0.02 seconds	

Random Forest Classifier (class_weight = 'balanced')		
Metrics	Train Set Results	Test Set Results
Accuracy Score	99.99%	88.56%
F1 Score	99.97%	0.77%
Precision Score	99.98%	68.42%
Recall Score	99.96%	0.38%
ROC AUC Score	99.99%	63.65%
Training Time	12.92 seconds	
Prediction Time	0.92 seconds	

LightGBM Classifier (is_unbalance = True)		
Metrics	Train Set Results	Test Set Results
Accuracy Score	66.41%	63.85%
F1 Score	32.91%	27.03%
Precision Score	21.34%	17.58%
Recall Score	71.92%	58.45%
ROC AUC Score	76.17%	66.24%
Training Time	1.27 seconds	
Prediction Time	0.12 seconds	

XGBoost Classifier (scale_pos_weight = 7.72)		
Metrics	Train Set Results	Test Set Results
Accuracy Score	61.27%	61.43%
F1 Score	27.33%	27.02%
Precision Score	17.40%	17.25%
Recall Score	63.57%	62.32%
ROC AUC Score	67.33%	66.40%
Training Time	11.58 seconds	
Prediction Time	0.09 seconds	

Support Vector Classifier (class_weight = 'balanced')		
Metrics	Train Set Results	Test Set Results
Accuracy Score	64.88%	64.96%
F1 Score	26.46%	26.58%
Precision Score	17.41%	17.49%
Recall Score	55.17%	55.36%
ROC AUC Score	64.65%	65.12%
Training Time	7658.75 seconds	
Prediction Time	242.23 seconds	

Fig 22: Model Evaluation Metrics

7.2 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.

Hyperparameter Tuning for the best 3 models in our project :

1. Logistic Regression: Logistic regression does not really have any critical hyperparameters to tune.

Sometimes, we can see useful differences in performance or convergence with different **solvers** (solver).

Solver in ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'].

Regularization (penalty) : Used to specify the norm used in the penalization.

Penalty in ['none', 'l1', 'l2', 'elasticnet']

Note: Not all solvers support all regularization terms.

The **C** parameter controls the penality strength, which can also be effective.

C in [0.5, 0.1, 0.05, 0.01. 0.005]

Hyperparameter tuned in our Logistic model:
Penalty : L1
Solver : saga
C : 0.05

Logistic Regression (class_weight = 'balanced')		
Metrics	**Train Set Results**	**Test Set Results**
Accuracy Score	62.50%	62.99%
F1 Score	26.62%	**27.01%**
Precision Score	17.16%	17.44%
Recall Score	59.40%	**59.77%**
ROC AUC Score	65.11%	**65.70%**

Training Time	5.86 seconds
Prediction Time	0.008 seconds

2. XGBoost Classifier

Important Parameters of XGBoost:

- **min_child_weight** [default=1]: Defines the minimum sum of weights of all observations required in a child. Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
- **max_depth** [default=6]:The maximum depth of a tree, same as GBM. Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
- **gamma** [default=0]: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.
- **subsample** [default=1]: Denotes the fraction of observations to be randomly samples for each tree. Lower values make the algorithm more conservative and prevent overfitting but too small values might lead to under-fitting.
- **scale_pos_weight** [default=1]: Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative instances) / sum(positive instances).
- **objective** [default=reg:linear]:This defines the loss function to be minimized. Mostly used values are:
  binary:logistic –logistic regression for binary classification, returns predicted probability (not class)
  multi:softmax –multiclass classification using the softmax objective, returns predicted class (not probabilities)
  multi:softprob –same as softmax, but returns predicted probability of each data point belonging to each class.
- **learning_rate:** using a smaller learning rate and increasing the number of iterations may improve accuracy.
- **eval_metric** [default according to objective]: The metric to be used for validation data. The default values are rmse for regression and error for classification.

Hyperparameter tuned in our XGBoost model:

```
XGBClassifier(random_state = 0, silent = 0, scale_pos_weight = weight,
```

objective = 'binary:logistic', eval_metric = 'auc')

{'colsample_bytree': 0.95,
 'gamma': 0.5,
 'learning_rate': 0.01,
 'max_depth': 6,
 'min_child_weight': 12,
 'n_estimators': 1183,
 'reg_alpha': 0.05,
 'subsample': 0.9}

XGBoost Classifier (scale_pos_weight = 7.72)		
Metrics	**Train Set Results**	**Test Set Results**
Accuracy Score	63.68%	62.87%
F1 Score	29.24%	**27.51%**
Precision Score	18.82%	17.72%
Recall Score	65.49%	**61.51%**
ROC AUC Score	70.28%	**66.79%**
Training Time	163.74 seconds	
Prediction Time	0.77 seconds	

3. LightGBM Classifier:

- **boosting_type** (*string, optional (default = 'gbdt')*): 'gbdt', traditional Gradient Boosting Decision Tree
- **num_leaves:** the number of leaf nodes to use. Having a large number of leaves will improve accuracy, but will also lead to overfitting.
- **min_child_samples**: the minimum number of samples (data) to group into a leaf. The parameter can greatly assist with overfitting: larger sample sizes per leaf will reduce overfitting (but may lead to under-fitting).
- **max_depth:** controls the depth of the tree explicitly. Shallower trees reduce overfitting.
- **is_unbalance or scale_pos_weight**: the weight can be calculated based on the number of negative and positive examples.
- **learning_rate:** using a smaller learning rate and increasing the number of iterations may improve accuracy.

Hyperparameter tuned in our LGBMClassifier model:

LGBMClassifier (objective = 'binary', random_state = 0, n_jobs = -1, is_unbalance = True)

{'boosting_type': 'gbdt',

'learning_rate': 0.06475,
'max_depth': 178,
'min_child_samples': 22,
'n_estimators': 128,
'num_leaves': 23}

LightGBM Classifier (is_unbalance = True)		
Metrics	**Train Set Results**	**Test Set Results**
Accuracy Score	62.54%	62.07%
F1 Score	28.77%	**27.19%**
Precision Score	18.39%	17.42%
Recall Score	66.04%	**61.81%**
ROC AUC Score	69.70%	**66.58%**
Training Time	2.60 seconds	
Prediction Time	0.34 seconds	

Final Model Chosen: As we can see from the above results, LGBM and XGBoost are giving better results after hyperparameter tuning. But the training and prediction time of XGBoost is far more than that of LGBM. So XGBoost is not scalable and not suitable for production because it will increase costs and time while giving similar results as LGBM which takes only 2-3 seconds to train on this large dataset thus saving costs and time while giving the best results.

ROC Curve:

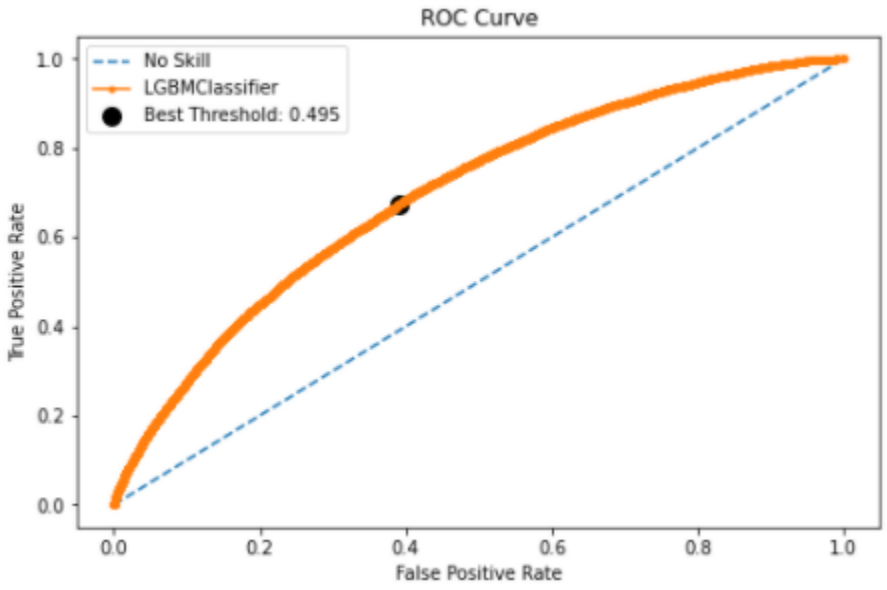

Fig 23: ROC Curve for LGBM

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. 0.495, following are the results of the final Model chosen to put in production:

LightGBM Classifier (in_unbalance = True)		
Metrics	**Train Set Results**	**Test Set Results**
Accuracy Score	61.68%	61.15%
F1 Score	28.73%	**27.22%**
Precision Score	18.25%	17.33%
Recall Score	67.42%	**63.42%**
ROC AUC Score	69.70%	**66.58%**

Confusion Matrix

15694	**10091**
1220	**2116**

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.

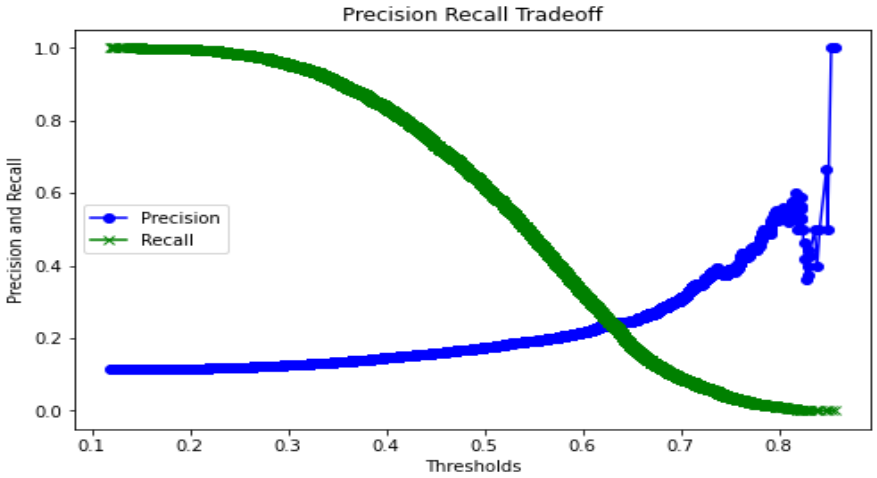

Fig 24: Precision Recall Tradeoff

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 70% and 75%.

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: (59.7 %, 63.6 %).

This means that we are 95% confident that the recall of the model will lie between 59.7% to 63.6%.

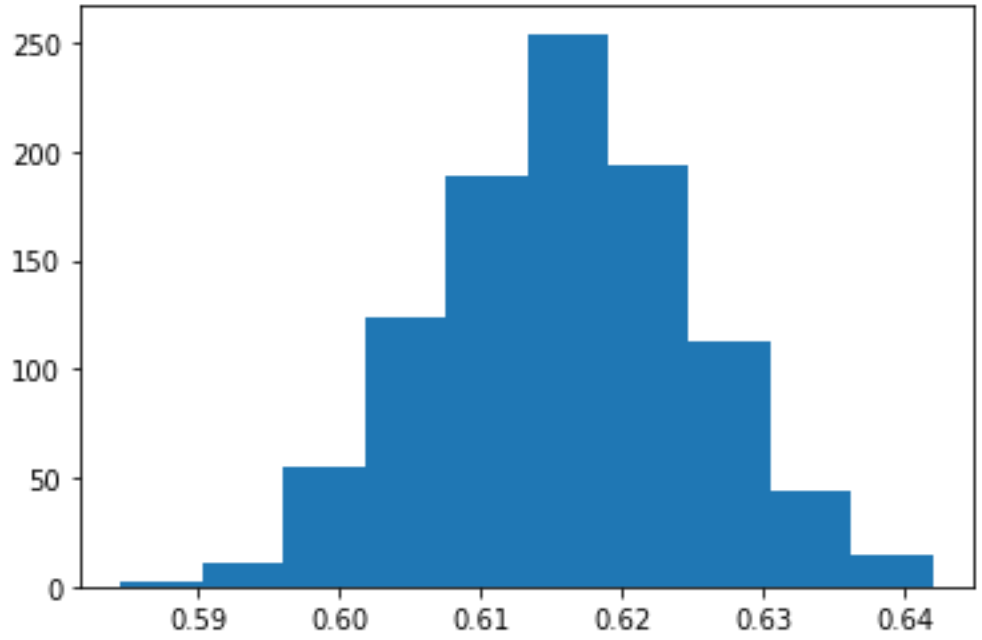

Fig 25: Histogram of Recall Scores over various samples of the dataset

CHAPTER 8

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 LIME:

Local Interpretable Model-agnostic Explanations (LIME), this is a model agnostic technique to generate local explanations to the model. The core idea behind the technique is quite intuitive. Suppose we have a complex classifier, with a highly non-linear decision boundary. But if we zoom in and look at a single prediction, the behavior of the model in that locality can be explained by a simple interpretable model (mostly linear). LIME uses a local surrogate model trained on perturbations of the data point we are investigating for explanations. This ensures that even though the explanation does not have global fidelity (faithfulness to the original model) it has local fidelity.

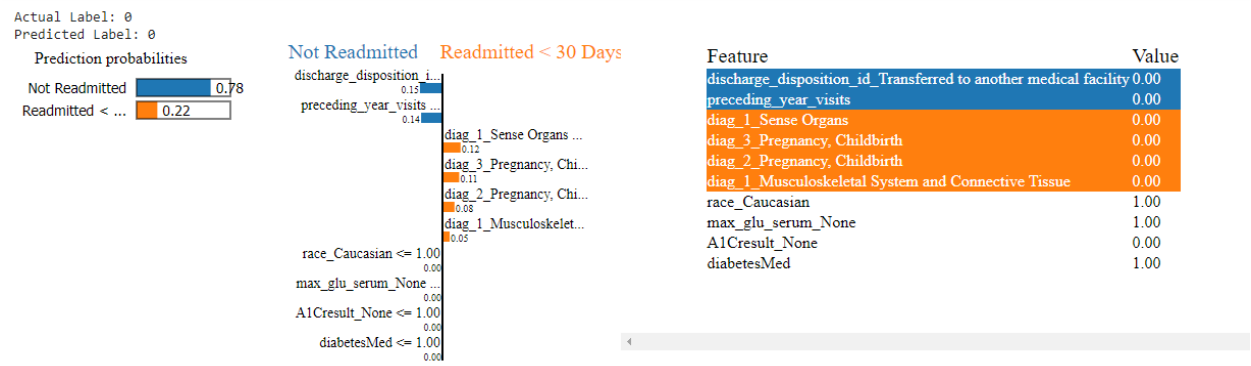

Fig 26: LIME Interpretation for a correctly predicted negative case

Fig 27: LIME Interpretation for a correctly predicted positive case

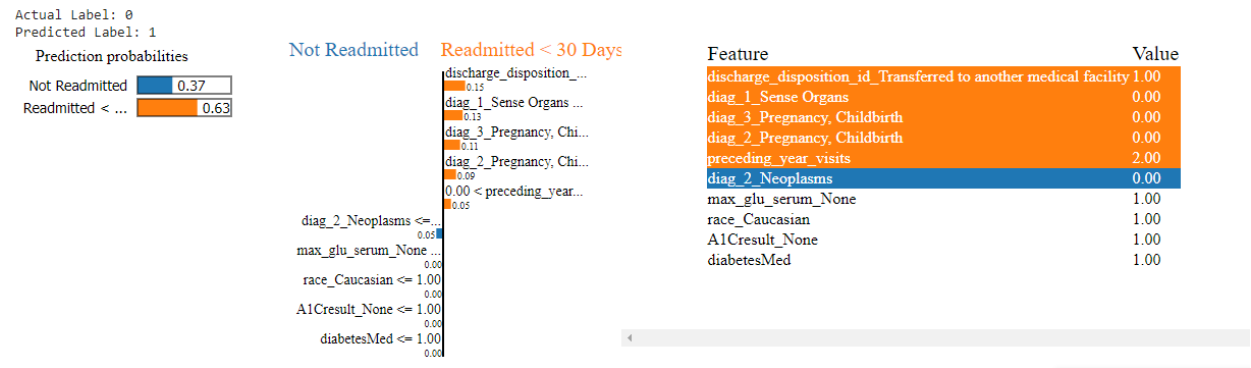

Fig 28: LIME Interpretations for a wrongly predicted case

Some Observations:

- We can see that preceding year visits is driving the model to predict readmission. More the visits higher the probability of readmission.
- Same goes for the discharge disposition id. If the person is discharged to another medical facility, the probability of getting readmitted increases.

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.

Feature Importance with SHAP

This basically takes the average of the SHAP value magnitudes across the dataset and plots it as a simple bar chart.

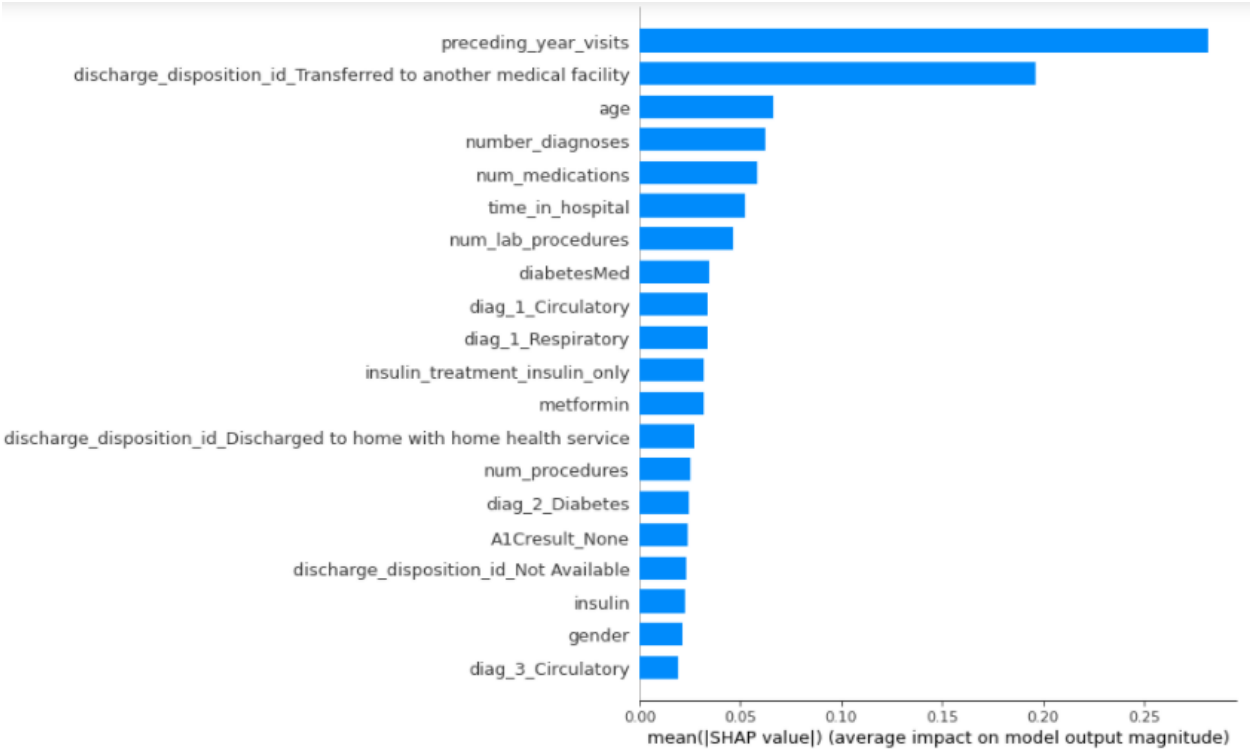

Fig 29: SHAP Feature Importance

The feature importance of the model gave us an idea of the factors that were potentially driving patients to be readmitted within 30 days. The most important feature for our selected model, as well as all other models, was the number of previous year visits. The higher the number, the higher the probability of the patient being readmitted. This same relationship could also be seen with number of diagnoses and time in hospital which were the 3 other numerical features in the top 10.

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.

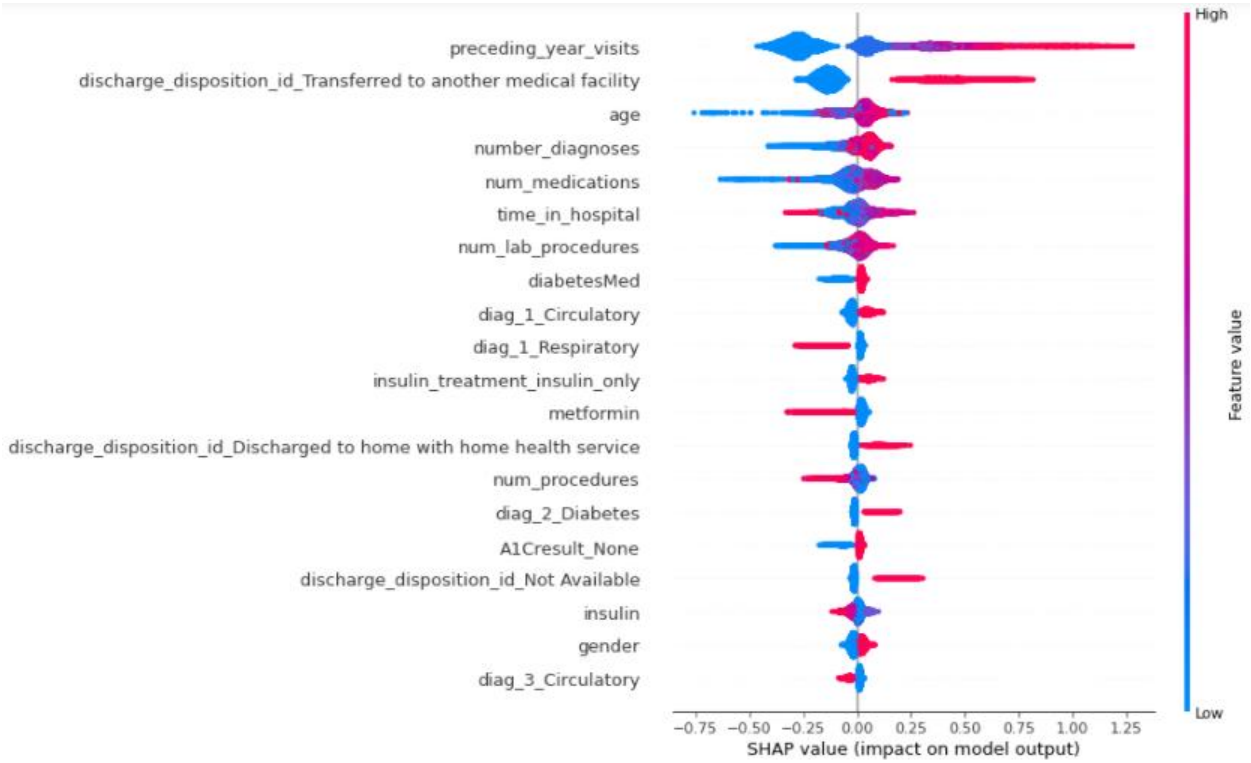

Fig 30: SHAP Summary Plot

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.
- If Metformin and/or insulin is not being given or the dosage is low.
- If secondary diagnosis was coming to be Diabetes.
- If A1C test was not performed.

CHAPTER 9

Business Recommendations & Future enhancements

9.1 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 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 (without any pre-processing) 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. Extending this research, we plan to further investigate the performance of classifiers with the goal to improve the accuracy of prediction of readmission risk. Further research is also warranted to explore the relevant feature space particularly with the variability of findings across research studies. The latter is particularly important as it can translate to proactive processes and policies aimed at addressing the factors that significantly influence hospital readmissions. 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.

9.2 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.

9.3 Limitations/ Challenges

Databases of clinical data contain valuable but heterogeneous and difficult data in terms of missing values, incomplete or inconsistent records, and high dimensionality understood not only by number of features but also their complexity. Additionally, analysing external data is more

challenging than analysis of results of a carefully designed experiment or trial, because one has no impact on how and what type of information was collected. Nonetheless, it is important to utilize these huge amounts of data to find new information/knowledge that is possibly not available anywhere. The number of observations belonging to readmitted within 30 days is significantly lower than those belonging to the other classes, which would result in incorrect predictions. Besides other factors mentioned above, the dataset does not include many important factors such as access to care, which has been shown in one study to account for 58% of variation readmission rates. There may be many other factors depending on situation that could be affecting readmissions. Also, the codes for diagnoses provided are according to the ICD 9 standards which needs to be updated according to the ICD 10 standards.

9.4 Future Scope

Geographical factor can also be included as a feature like whether a patient is from urban or a rural region. This would allow determining if the readmission factors differ based on patient geographical location or if similar traits are observed nationwide. In addition, this would strengthen both urban and rural models while assessing the importance of age categorization.

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.

Living with diabetes is challenging and distressful. Diabetic patient's condition cannot be understood only from his medical charts. There is a need to collect and analyze both subjective and objective patient information in order to fully understand the occurrence of readmission of patients with diabetes. Subjective data can be captured by interviewing patients or by conducting surveys which will enrich the depth of patient information. The conversation between doctor and patient can also be collected and analyzed which could help to extract important features corresponding to patient's willpower and attitude by text-mining techniques. This information might improve the intelligent models to identify patients at high risk of readmission.

Given more time, we could run stacked models and neural networks to see if the AUC and recall is improved or not. We can also dive deeper into the most important features of the models to see which particular categories of the features are affecting the classification. We can also try to change the classification threshold for some models to see if they improve the performance, especially reducing the false positives to improve the precision score.

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