DIABETIC PATIENT'S READMISSION PREDICTION

Post Graduate Program in Data Science Engineering

Location: **Chennai** Batch: **PGPDSE-FT Chennai Sep'21**

Submitted by

1. UMAIR AHMED E F
2. TEJA LAKKADASU
3. VISHAL KARTHICK R
4. MANNE VASU RAJENDRA
5. YOGESHWARAN V

Mentored by

Mr. SRIKAR MUPPIDI

Table of Contents

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

Visualization: Python (Matplotlib and Seaborn)

Models: Base model

INTRODUCTION

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%).

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.

Impact on business

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.

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.

Variable information/Data description:

S.no	Feature name	Description
1.	Encounter ID	Unique identifier of an encounter
2.	Patient Number	Unique identifier of a patient
3.	Race	Values: Caucasian, Asian, African American, Hispanic, and other
4.	Gender	Values: male, female, and unknown/invalid

5.	Age	Grouped in 10-year intervals: [(0, 10), (10, 20), ..., (90, 100)]
6.	Weight	Weight in pounds
7.	Admission Type	Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, new-born, and not available
8.	Discharge disposition	Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available
9.	Admission source	Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital
10.	Time in hospital	Integer number of days between admission and discharge
11.	Payer Code	Integer identifier corresponding to 23 distinct values, for example, Blue Cross\Blue Shield, Medicare, and self-pay
12.	Medical Speciality	Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family, general practice, and surgeon
13.	Number of Outpatient visits	Number of outpatient visits in the year preceding the encounter

14.	Number of lab procedures	Number of lab tests performed during encounter
15.	Number of procedures	Number of procedures (other than lab tests) performed during the encounter
16.	Number of Medications	Number of distinct generic names administered during the encounter
17.	Number of emergency visits	Number of emergency visits of the patient in the year preceding the encounter
18	Number of inpatient visits	Number of inpatient visits of the patient in the year preceding the encounter
19.	Diagnosis 1	The primary diagnosis (coded as first three digits of ICD9) 848 distinct values
20.	Diagnosis 2	Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
21.	Diagnosis 3	Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct
22.	Number of Diagnoses	Number of diagnoses entered in the system

23.	Glucose serum test result	Indicates the range of the result or if the test was not taken. Values: “>200,” “>300,”, “normal,” and “none” if not measured
24.	A1c test result	Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>7” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured
25.	Change of medications	Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”
26.	Diabetics medication	Indicates if the there was any diabetic medication prescribed. Values: “yes” and “no”
27.	24 features for medication	For the generic names: metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, and metformin-pioglitazone, the feature indicates whether the drug was prescribed or there was a change in the dosage. Values: “up” if the dosage was increased during the encounter, “down” if the dosage was decreased, “steady” if the dosage did not change, and “no” if the drug was not prescribed
28.	Readmitted	Days to inpatient readmission. Values: “<30” if the patient was readmitted in less than 30 days, “>30” if the patient was readmitted in more than 30 days, and “No” for no record of readmission

Data Preprocessing:

Pre-processing Data Analysis:

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

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

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 (‘? ‘).

```python
dfnull = pd.DataFrame({'Missing_Values':dfm,'Percentanges':dfper},index=df.columns)
dfnull
```

	Missing_Values	Percentanges
encounter_id	0	0.000000
patient_nbr	0	0.000000
race	2273	2.233555
gender	3	0.002948
age	0	0.000000
weight	98569	96.858479
admission_type_id	0	0.000000
discharge_disposition_id	0	0.000000
admission_source_id	0	0.000000
time_in_hospital	0	0.000000
payer_code	40256	39.557416
medical_specialty	49949	49.082208
num_lab_procedures	0	0.000000
num_procedures	0	0.000000
num_medications	0	0.000000
number_outpatient	0	0.000000
number_emergency	0	0.000000
number_inpatient	0	0.000000
diag_1	21	0.020636
diag_2	358	0.351787
diag_3	1423	1.398306
number_diagnoses	0	0.000000
max_glu_serum	0	0.000000
A1Cresult	0	0.000000
metformin	0	0.000000

As the data contains more missing values for medical_specality, weight, payer_code we are dropping these attributes.

```
df.drop('medical_specialty',1,inplace=True)
```

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

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

Combining categories in admission_type_id

Admission Source Id:

Referral	• Physician Referral • Clinic Referral • HMO Referral (Health Maintenance Organization)

Transferred from another health care 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 ina 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 aftercare • Discharged/transferred/referred another institution for outpatient services

	• Discharged/transferred to another rehab fac including rehab units of a hospital. • Discharged/transferred to a long-term care hospital. • Discharged/transferred to a nursing facility certified under Medicaid but not certified under Medicare. • Discharged/transferred to a federal health care facility. • Discharged/transferred/referred to a psychiatric hospital 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 service • Discharged/transferred to home under care of Home I V provider
Still patient/referred to this institution	• Admitted as an inpatient to this hospital • Still patient or expected to return for outpatient services • Discharged/transferred within this institution to Medic are approved swing bed • Discharged/transferred/referred to this institution for outpatient 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

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

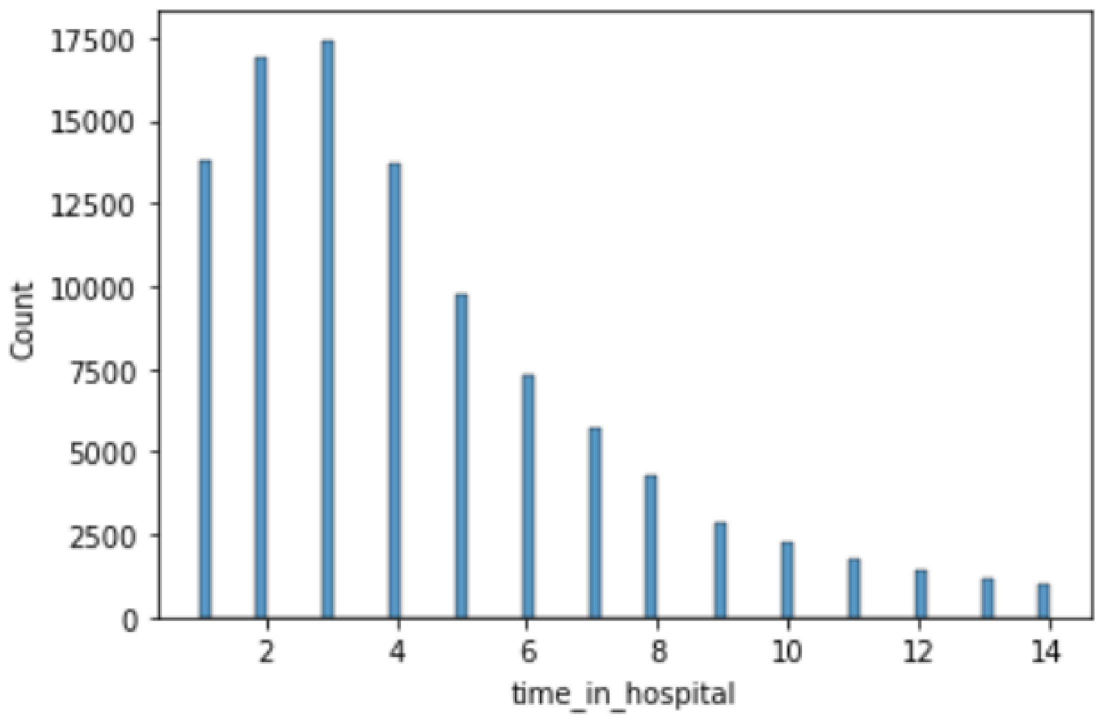
- Circulatory (390–459,785)
- Respiratory (460–519,786)
- Digestive (520–579,787)
- Diabetes (250.xx)
- Injury (800–999)
- Musculoskeletal (710–739)
- Genitourinary (580–629)
- Neoplasms (140–239)
- Other- The codes which are not present in the above list has been classified here

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

    x = float(x)

    if (x >= 390 and x <= 459) or np.floor(x) == 785:
        return 'Circulatory'
    elif (x >= 460 and x <= 519) or np.floor(x) == 786:
        return 'Respiratory'
    elif (x >= 520 and x <= 579) or np.floor(x) == 787:
        return 'Digestive'
    elif np.floor(x) == 250:
        return 'Diabetes'
    elif x >= 800 and x <= 999:
        return 'Injury'
    elif x >= 710 and x <= 739:
        return 'Musculoskeletal'
    elif (x >= 580 and x <= 629) or np.floor(x) == 788:
        return 'Genitourinary'
    elif x >= 140 and x <= 239 or np.floor(x) in [780, 781, 784] or x >= 790 and x <=799 or x>=240 and x<=249 or x>=251 and x<=2
        return 'Neoplasms'
    else:
        return 'Others'
```

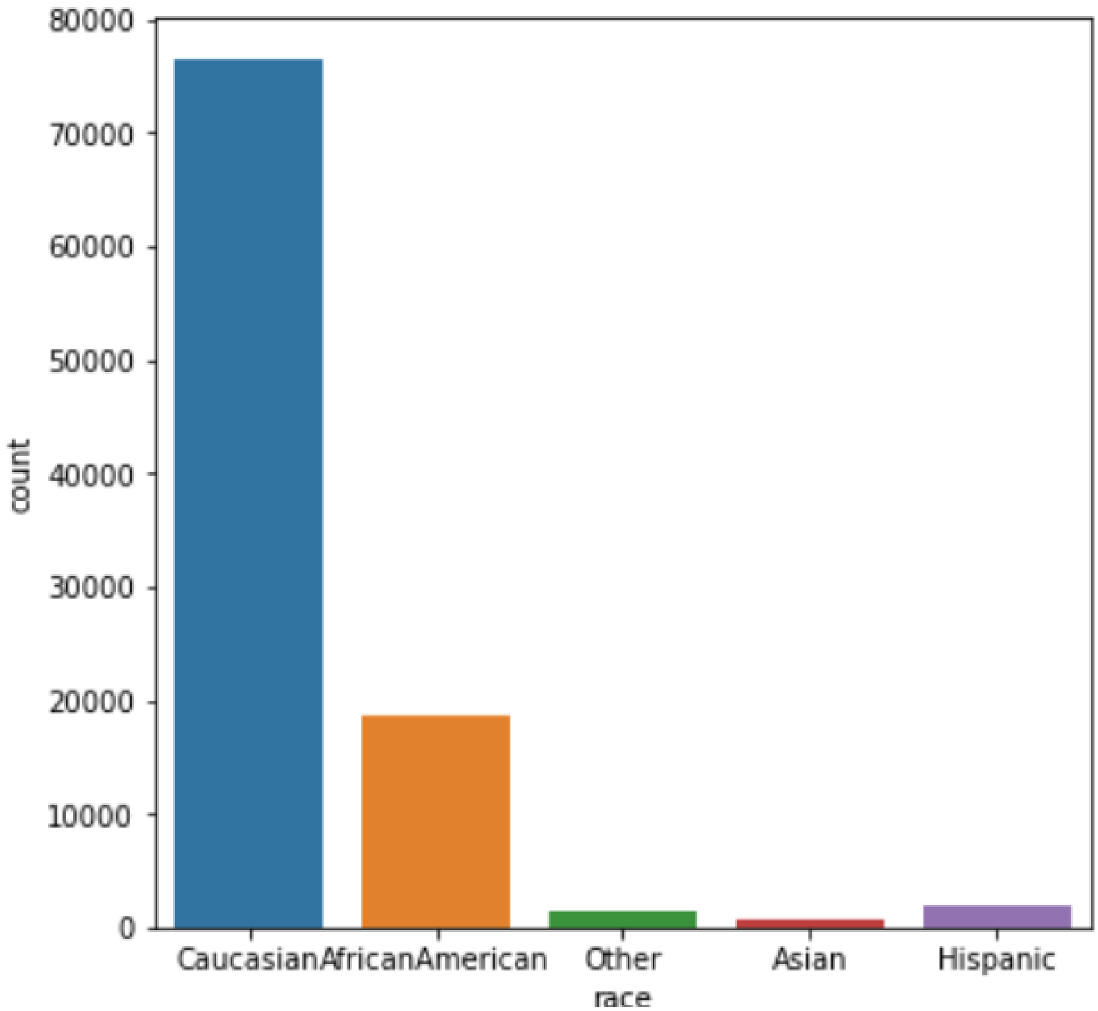
Univariate analysis:

Time in hospital

It is a continuous numerical column. The maximum frequency of people readmitted has been observed in 1-4 days.

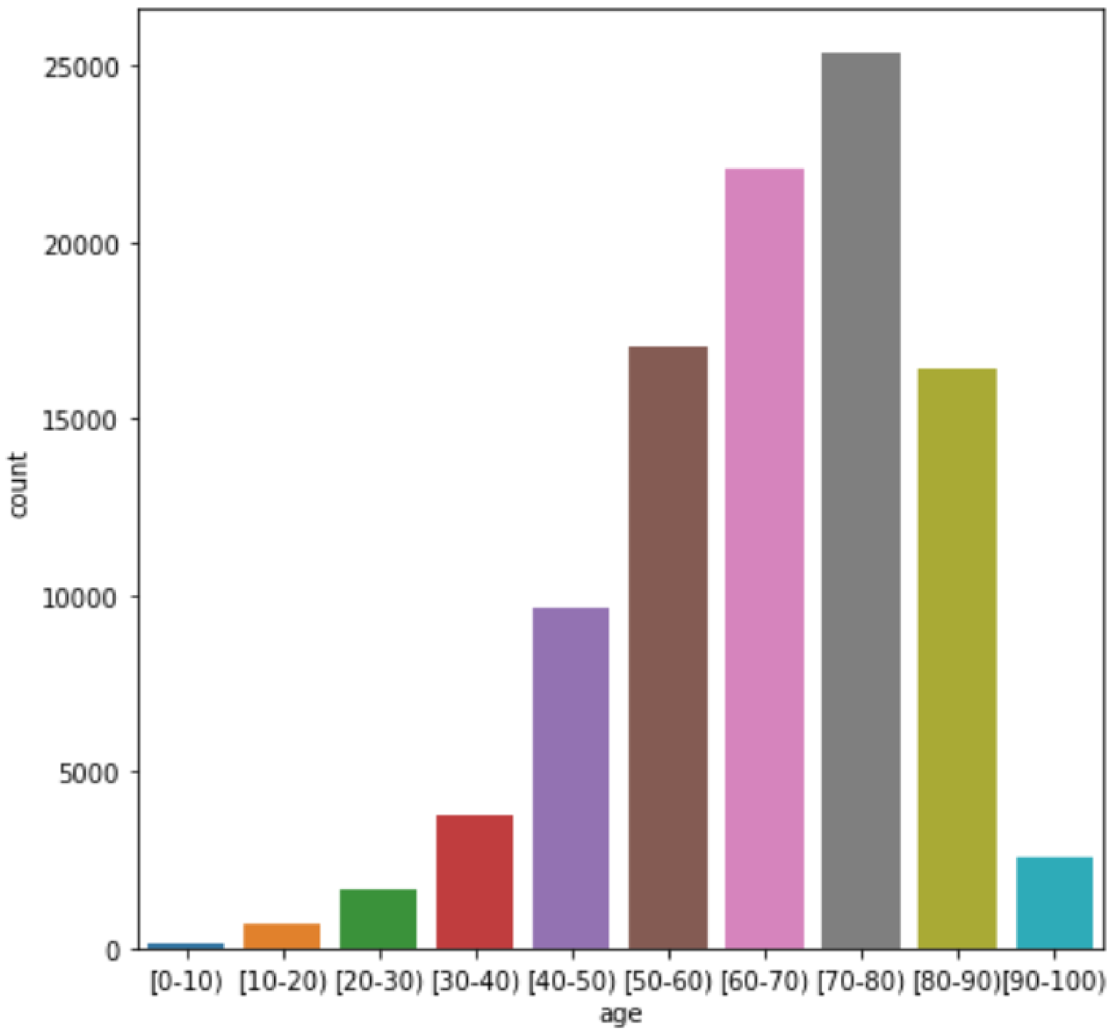


Race: Majority of the population present in the dataset is Caucasians followed by African-American



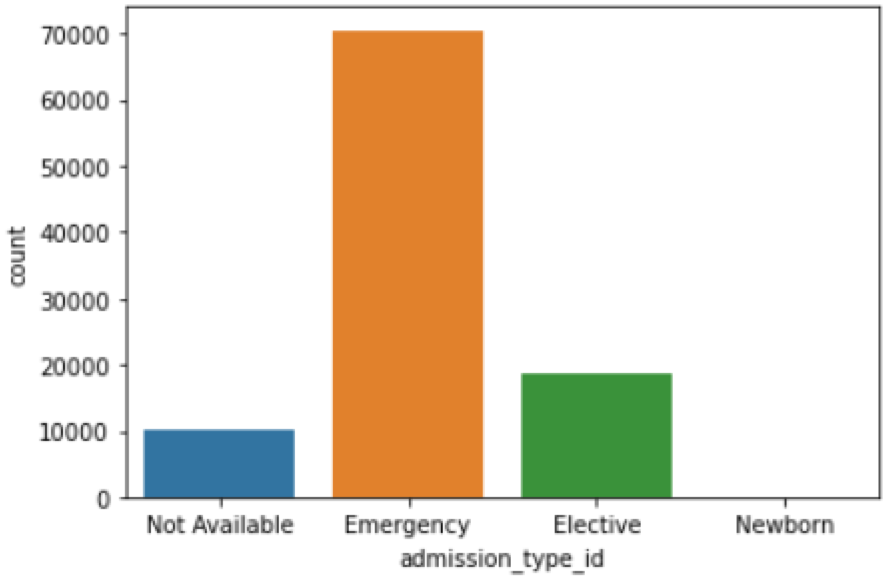
Age:

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



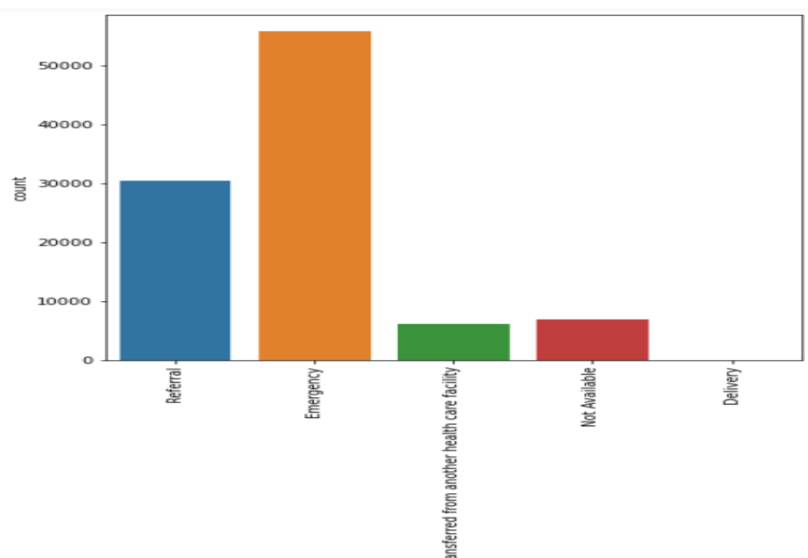
Admission type id

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



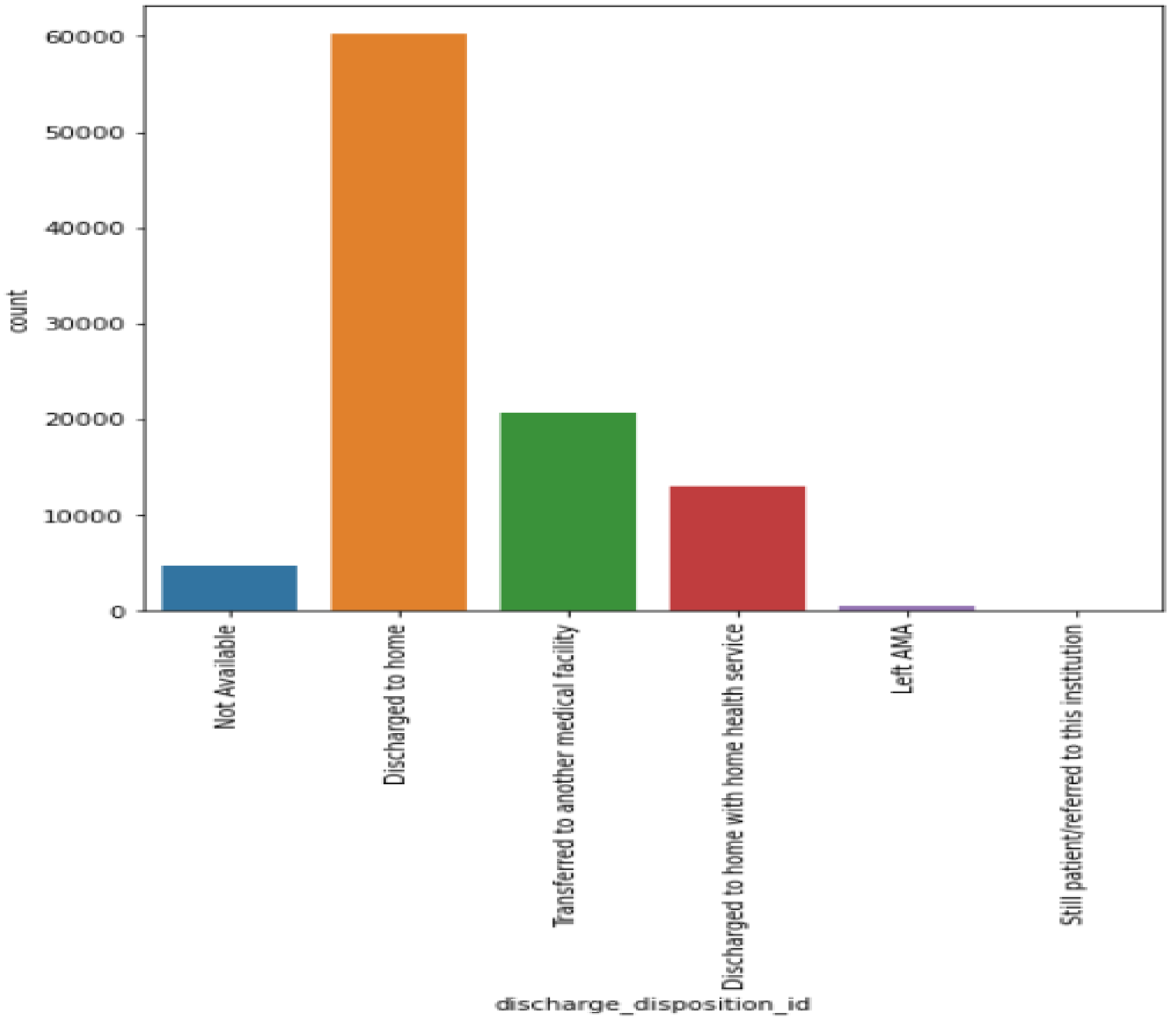
Admission Source ID

The Admission source types have been grouped into 5 categories. The Emergency care patients are the most prevalent ones.



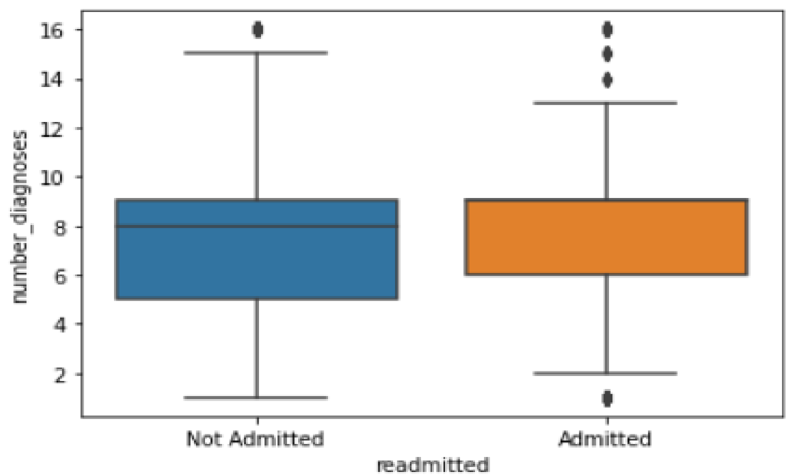
Discharge Disposition:

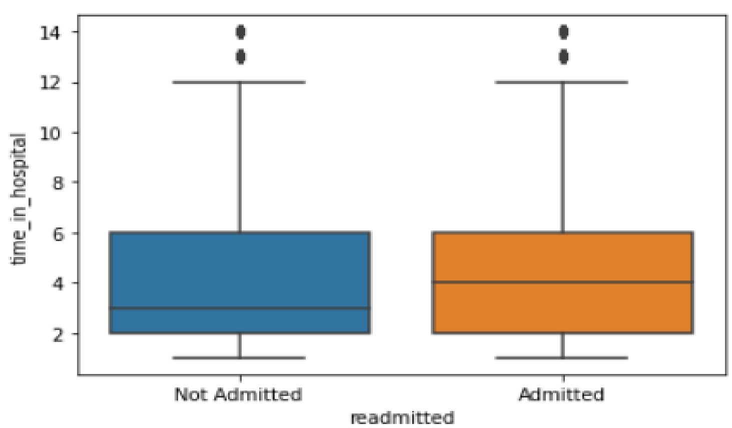
Most of the patients were discharged to home and to a short-term hospital.



Bivariate:

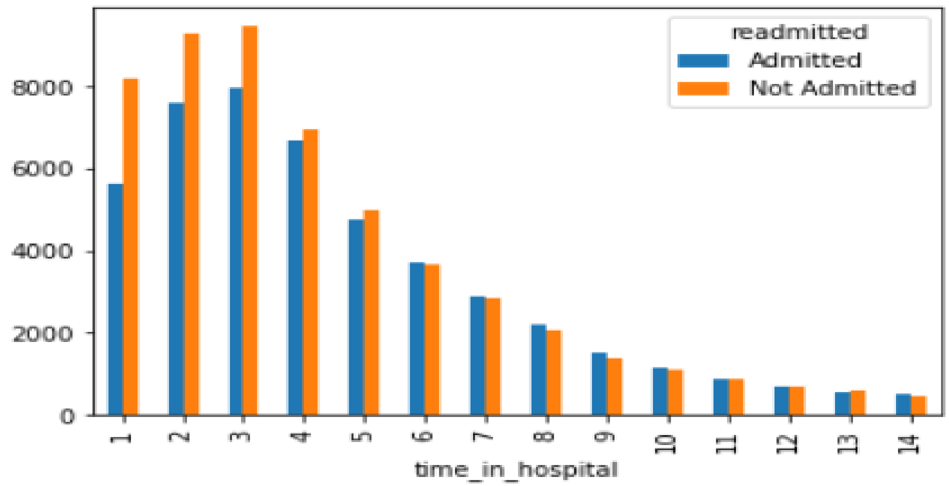
Analysed the impact of the entire continuous variable on the target variable from the box plot and cross tabs. Plotted only a few samples here. There are few outliers in some of the variables. However, all three categories of readmitted categories seem to show similar characteristics.





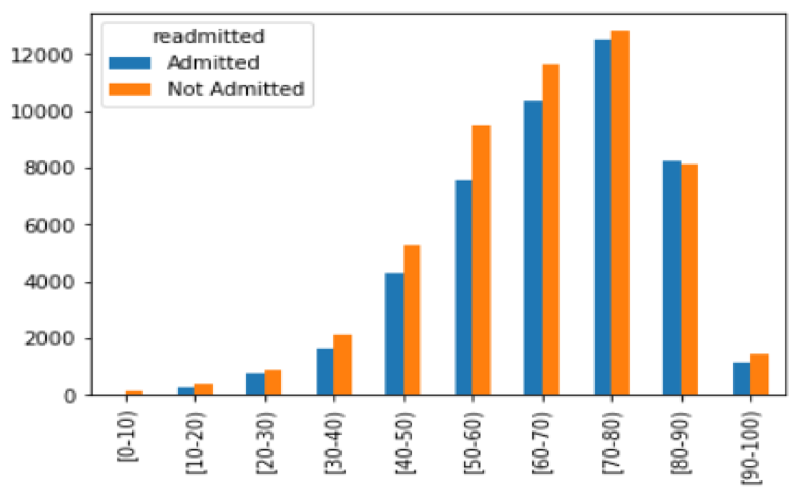
Time in hospital vs Distribution of Readmission

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



Age of Patient vs Readmission

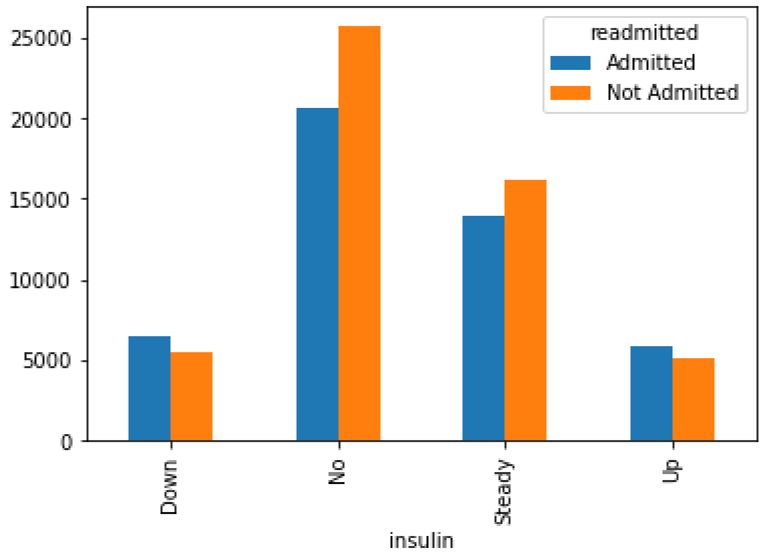
Patients above the age of 60 are highly prone to diabetics



Insulin, Readmitted (count):

The level of insulin was steady for nearly 25000 patients overall.

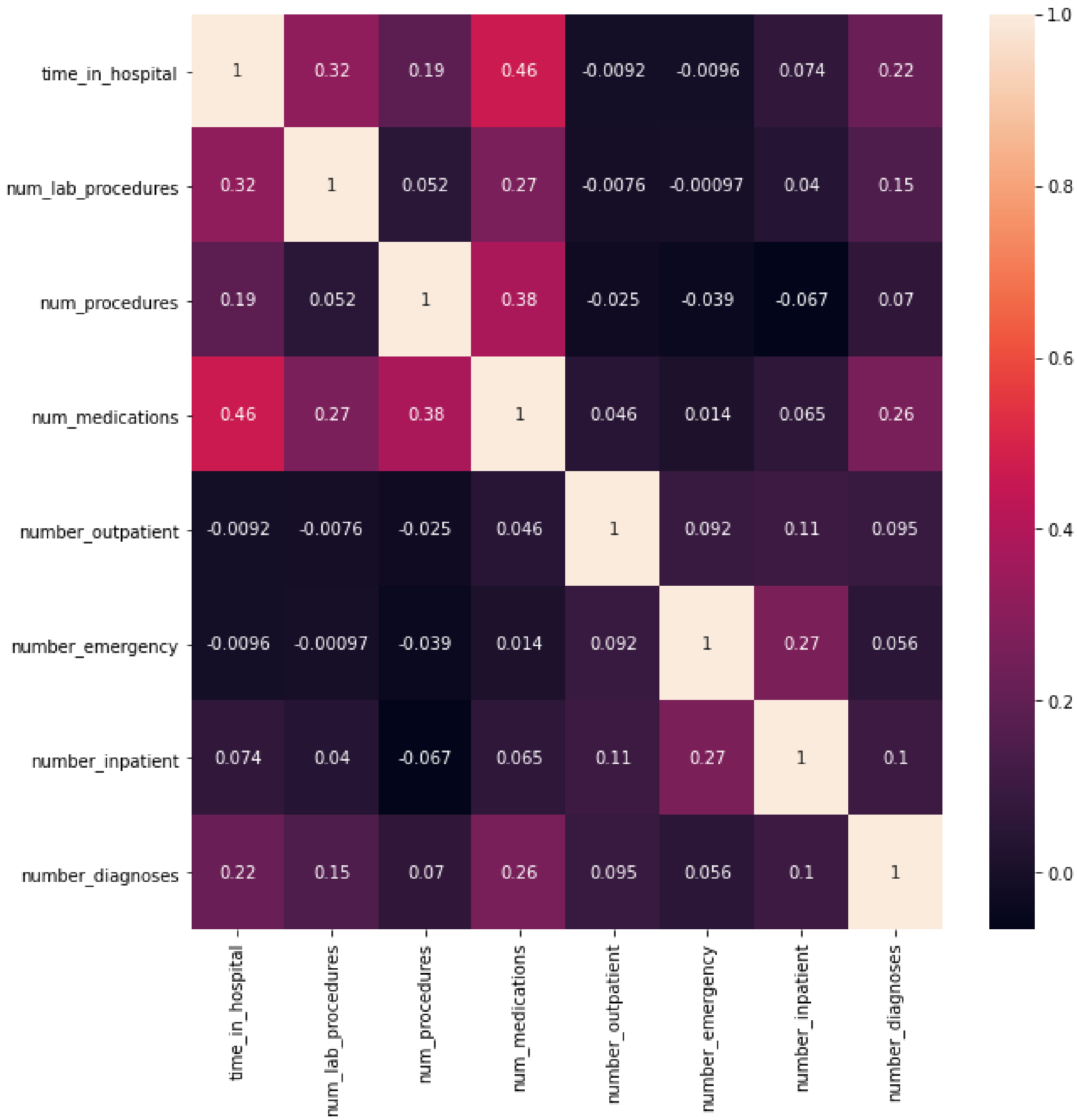
The patients who were not administered with insulin are highly likely not to be readmitted



Heat map:

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

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



Encoding Attributes:

For understanding the attributes information to the machine we are encoding some attributes we are creating dummy attributes which is technically called as one hot encoding.

```
df = pd.get_dummies(df,columns=['race', 'admission_type_id', 'discharge_disposition_id',
                                'admission_source_id', 'max_glu_serum','A1Cresult','metformin',
                                'repaglinide','nateglinide','chlorpropamide','glimepiride',
                                'acetohexamide','glipizide','glyburide','tolbutamide','pioglitazone',
                                'rosiglitazone','acarbose','miglitol','troglitazone','tolazamide',
                                'insulin','glyburide-metformin','glipizide-metformin','glimepiride-pioglitazone',
                                'glimepiride-pioglitazone','metformin-rosiglitazone','metformin-pioglitazone' ],drop_first=True)
```

Target variable:

```
df['readmitted']= df['readmitted'].replace( {'Not Admitted': 0, 'Admitted': 1} )
```

MACHINE LEARNING BASE MODEL

After cleaning the initial dataset and post dummy encoding the variables, we have 106 columns with 99337 data rows. So, we proceeded to build a few base models using Logistic Regression, Random Forest on this data without scaling or transformation or hyperparameter tuning until the model performance scores are in acceptable range. Below is the data frame consisting of all the scores.

In this section, we will first compare the performance of the following 2 machine learning models using default hyperparameters:

- LOGISTIC REGRESSION
- RANDOM FOREST

Logistic Regression

Logistic regression is a traditional machine learning model that fits a linear decision boundary between the positive and negative samples. This linear function is then passed through a sigmoid function to calculate the probability of the positive class. Logistic regression is an excellent model to use when the features are linearly separable. One advantage of logistic regression is the model is interpretability — i.e., we know which features are important for predicting positive or negative. One thing to consider is that the modelling is sensitive to the scaling of the features, so that is why we scaled the features above in such a way that scaled variable distributions to satisfy the assumptions of logistic regression.

```python
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report,precision_score,recall_score,f1_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,random_state=1)
dt = LogisticRegression()
dt.fit(X_train,y_train)
print("Training Accuracy")
print(dt.score(X_train,y_train))
print("Testing Accuracy")
print(dt.score(X_test,y_test))
predicted = dt.predict(X_test)
print(confusion_matrix(y_test,predicted))
print(classification_report(y_test,predicted))
print(precision_score(y_test,predicted))
print(recall_score(y_test,predicted))
print(f1_score(y_test,predicted))
```

```
Training Accuracy
0.6197762649586631
Testing Accuracy
0.6197402858868533
[[7977 2401]
 [5154 4336]]
              precision    recall  f1-score   support

           0       0.61      0.77      0.68     10378
           1       0.64      0.46      0.53      9490

    accuracy                           0.62     19868
   macro avg       0.63      0.61      0.61     19868
weighted avg       0.62      0.62      0.61     19868

0.643609915392608
0.45690200210748155
0.5344179453996426
```

Random forest

One disadvantage of decision trees is that they tend overfit very easily by memorizing the training data. As a result, random forests were created to reduce the overfitting. In random forest models, multiple trees are created and the results are aggregated. The trees in a forest are decorrelated by using a random set of samples and random number of features in each tree. In most cases, random forests work better than decision trees because they are able to generalize more easily.

```python
# instantiate the 'RandomForestClassifier'
# pass the required number of trees in the random forest to the parameter, 'n_estimators'
# pass the 'random_state' to obtain the same samples for each time you run the code
rf_classification = RandomForestClassifier(n_estimators = 15, random_state = 10)

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

# predict the attrition for test set
y_pred = rf_model.predict(X_train)

# generate a classification report
print(classification_report(y_train, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     42147
           1       1.00      0.99      1.00     37322

    accuracy                           1.00     79469
   macro avg       1.00      1.00      1.00     79469
weighted avg       1.00      1.00      1.00     79469
```

FUTURE WORK:

- Treating the outliers and treating multicollinearity with VIF.
- Scaling and Transformation for modelling.
- Revisiting Feature engineering /Feature selection process.
- Implementing various classification algorithms for the best model selection
- Use Ensemble techniques to improve the model performance
- Hyperparameter Tuning.
- Robust Model Evaluation.
- Improving Precision and Recall Scores for Minority class.