**Final Project Report**

**Title** **-** **Diabetic patient readmission prediction.**

#### Group 2

**Mentor:** Vidya K

**Members:** V Keerthi Vikram, Sarvesh Mankar, Kajal Chopda, Nishad Kharwade, Vidhushi Vanjari.

Contents

[Problem Statement 1](#_Toc46763520)

[Literature Survey 1](#_Toc46763521)

[Introduction 1](#_Toc46763522)

[Descriptive Statistics 1](#_Toc46763523)

[EDA 1](#_Toc46763524)

[Missing value imputation 1](#_Toc46763525)

[Statistical test 1](#_Toc46763526)

[Feature Engineering 1](#_Toc46763527)

[Correlation 1](#_Toc46763528)

[Pearson Correlation 1](#_Toc46763529)

[Cramer's V 1](#_Toc46763530)

[Multicollinearity 1](#_Toc46763531)

[Variance Inflation Factor 1](#_Toc46763532)

[Skewness and Transformation 1](#_Toc46763533)

[Boxcox Transformation 1](#_Toc46763534)

[Standardization 1](#_Toc46763535)

[Standard Scaler 1](#_Toc46763536)

[Encoding 1](#_Toc46763537)

[Mean Encoding 1](#_Toc46763538)

[One-Hot Encoding 1](#_Toc46763539)

[Visualizations 1](#_Toc46763540)

[Balancing the Imbalanced dataset 1](#_Toc46763541)

[Model Building 1](#_Toc46763542)

[Models Used 1](#_Toc46763543)

[Logistic Regression 1](#_Toc46763544)

[Decision Tree Classifier 1](#_Toc46763545)

[Support Vector Classifier 1](#_Toc46763546)

[KNN 1](#_Toc46763547)

[Random Forest Classifier 1](#_Toc46763548)

[XGBoost Classifier 1](#_Toc46763549)

[LightGBM 1](#_Toc46763550)

[Scenario 1 1](#_Toc46763551)

[Scenario 2 1](#_Toc46763552)

[Scenario 3 1](#_Toc46763553)

[Business Inference 1](#_Toc46763554)

# Problem Statement

To help the hospital prioritize attention towards certain diabetic patients who have a high probability of getting readmitted to the hospital based on past information from the data collected.

# Literature Survey

* “Diabetes 130 US hospitals dataset” by Weiyan Wu, Yao Jin, Jiwei Wang, Lingyun Mao, Han W.
* “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records” By Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios and John N. Clore.

# Introduction

Type 1 diabetes is a disease in which the body does not make enough insulin to control blood sugar levels. Type 1 diabetes was previously called insulin dependent diabetes or juvenile diabetes. Type 1 diabetes is an autoimmune condition. It happens when your body attacks your pancreas with antibodies. The organ is damaged and doesn't make insulin.

When you have type 2 diabetes, your pancreas usually creates some insulin. But either it’s not enough or your body doesn’t use it like it should. Insulin resistance, when your cells don’t respond to insulin, usually happens in fat, liver, and muscle cells. Type 2 diabetes is often milder than type 1. But it can still cause major health complications, especially in the tiny blood vessels in your kidneys, nerves, and eyes. Type 2 also raises your risk of heart disease and stroke. Type 2 diabetes used to be called non-insulin-dependent or adult onset diabetes. But it’s become more common in children and teens over the past 20 years, largely because more young people are overweight or obese.

# Descriptive Statistics

The shape of the dataset was checked which was found to be 1,01,763 rows and 50 columns.

Following this the data type of columns was found out.

It could be seen that there were 9 numerical and 41 categorical columns.

Upon observation it was found that 3 columns were pre encoded.

It was also observed that we had 24 columns related to the dosage Increase, Decrease, Steady or not given) for the drugs. Upon performing boxplot, we could also observe there were outliers.

Also, outliers could be observed looking at the difference in mean and median value.

Our target column is readmitted which is a categorical column.

# EDA

## Missing value imputation

Missing values were present in the dataset in Weight, Race, Diag\_1, Diag\_2, Diag\_3, Payer code, Medical Specialty.

Weight, Payer code, Medical Specialty columns were dropped due to above 40% missing values.

Missing values in Race was imputed using mode.

Diag\_1, Diag\_2, Diag\_3 had missing values replaced by No Diagnosis Done.

## Statistical test

Statistical tests such as chi2 contingency test was done on categorical columns and logistic regression Logit) was done on numerical columns and non-significant columns were dropped.

Some more columns were dropped due to Insufficient data based on domain knowledge.

Columns dropped are acarbose, acetohexamide, chlorpropamide, citoglipton, examide, glimepiride-pioglitazone, glipizide-metformin, glyburide-metformin, metformin-pioglitazone, metformin-rosiglitazone, miglitol, tolazamide, troglitazone, tolbutamide, nateglinide, glyburide.

## Feature Engineering

We combined num\_lab\_procedures and num\_procedures to get num\_ovrll\_prcdrs and combined number\_outpatient , number\_emergency and number\_inpatient to get num\_prvs\_vists.

We further generalized diag\_1, diag\_2 and diag\_3 columns based on ICD9 2008 code to strings.

We dropped the rows containing others subclass of gender column as it had very less data points and could have caused bias.

## Correlation

It was checked and none of the columns were found to be highly correlated with each other. Pearson correlation was used for numerical columns and Cramers\_v correlation was used for categorical columns

### Pearson Correlation

In statistics, the Pearson correlation coefficient also referred to as Pearson's r, the Pearson product-moment correlation coefficient (PPMCC), or the bivariate correlation is a statistic that measures linear correlation between two variables X and Y. It has a value between 1 and 1. A value of 1 is total positive linear correlation, 0 is no linear correlation, and 1 is total negative linear correlation.

Where,

*cov* is the covariance

*σX* is the standard deviation of X

*σY* is the standard deviation of Y

### Cramer's V

In statistics, Cramer's V is a measure of association between two nominal variables, giving a value between 0 and 1 (inclusive). It is based on Pearson's chi-squared statistic.

Cramer's V may also be applied to goodness of fit chi-squared models when there is a 1 × k table (in this case r = 1).

*V* = *φ*2 = *χ*2/*n*

*min*(*k* − 1,*r* − 1) *min*(*k* − 1,*r* − 1)

Where,

*φ* is the phi coefficient

*χ*2 is derived from Pearson's chi-squared test

*n* is the grand total observations

*k* is the number of columns

*r* is the number of rows

## Multicollinearity

It was checked for and the columns were found to not have multicollinearity.

### Variance Inflation Factor

In statistics, multicollinearity (also collinearity) is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy. In this situation, the coefficient estimates of the multiple regression may change erratically in response to small changes in the model or the data. Multicollinearity does not reduce the predictive power or reliability of the model as a whole, at least within the sample data set; it only affects calculations regarding individual predictors. That is, a multivariate regression model with collinear predictors can indicate how well the entire bundle of predictors predicts the outcome variable, but it may not give valid results about any individual predictor, or about which predictors are redundant with respect to others.

In statistics, the variance inflation factor VIF is the quotient of the variance in a model with multiple terms by the variance of a model with one term alone. It quantifies the severity of multicollinearity in an ordinary least squares regression analysis. It provides an index that measures how much the variance (the square of the estimate's standard deviation) of an estimated regression coefficient is increased because of collinearity.

=

where *Ri*2 is the coefficient of determination of the regression equation.

**New Shape**

Following this our new number of rows is 1,01,760 and our new number of columns is 26.

## Skewness and Transformation

We checked for skewness in the dataset and the numerical columns were found to have some skewness.

### Boxcox Transformation

A Box Cox transformation is a way to transform non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques; if your data isn’t normal, applying a Box-Cox means that you are able to run a broader number of tests.

At the core of the Box Cox transformation is an exponent, lambda (λ), which varies from 5 to 5. All values of λ are considered and the optimal value for your data is selected; The “optimal value” is the one which results in the best approximation of a normal distribution curve. The transformation of Y has the form:

if *λ* ≠ 0

*y*(*λ*) = *log*(*y*), *if* *λ* = 0

## Standardization

Following this the numerical columns were scaled down to the same scale using standard scaler.

### Standard Scaler

Standardize generally means changing the values so that the distribution standard deviation from the mean equals one. It outputs something very close to a normal distribution. Standard Scaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation. Standard Scaler results in a distribution with a standard deviation equal to 1. The variance is equal to 1 also, because *variance = standard deviation squared*. And 1 squared = 1. Standard Scaler makes the mean of the distribution 0. About 68% of the values will lie be between 1 and 1.

## Encoding

We have done mean encoding on the diagnosis columns as they have high cardinality. This was followed by one hot encoding wherever possible and the remaining areas mean encoding.

### Mean Encoding

from sklearn.base import BaseEstimator

class MeanEncoding(BaseEstimator):

""" In Mean Encoding we take the number

of labels into account along with the target variable

to encode the labels into machine comprehensible values """

def \_\_init\_\_(self, feature, C=0.1):

self.C = C

self.feature = feature

def fit(self, X\_train, y\_train):

df = pd.DataFrame(

{'feature': X\_train[self.feature], 'target': y\_train}).dropna()

self.global\_mean = df.target.mean()

mean = df.groupby('feature').target.mean()

size = df.groupby('feature').target.size()

self.encoding = (self.global\_mean \* self.C +

mean \* size) / (self.C + size)

def transform(self, X\_test):

X\_test[self.feature] = X\_test[self.feature].map(

self.encoding).fillna(self.global\_mean).values

return X\_test

def fit\_transform(self, X\_train, y\_train):

df = pd.DataFrame(

{'feature': X\_train[self.feature], 'target': y\_train}).dropna()

self.global\_mean = df.target.mean()

mean = df.groupby('feature').target.mean()

size = df.groupby('feature').target.size()

self.encoding = (self.global\_mean \* self.C +

mean \* size) / (self.C + size)

X\_train[self.feature] = X\_train[self.feature].map(

self.encoding).fillna(self.global\_mean).values

return X\_train

for f in cat\_cols:

me = MeanEncoding(f, C=0.01 \* len(X[f].unique()))

me.fit(X, y)

X = me.transform(X)

test = me.transform(test)

In Mean Encoding we take the number of labels into account along with the target variable to encode the labels into machine comprehensible values.

Mean encoding can embody the target in the label whereas label encoding has no correlation with the target.

In case of large number of features, mean encoding could prove to be a much simpler alternative.

A histogram of predictions using label & mean encoding show that mean encoding tend to group the classes together whereas the grouping is random in case of label encoding.

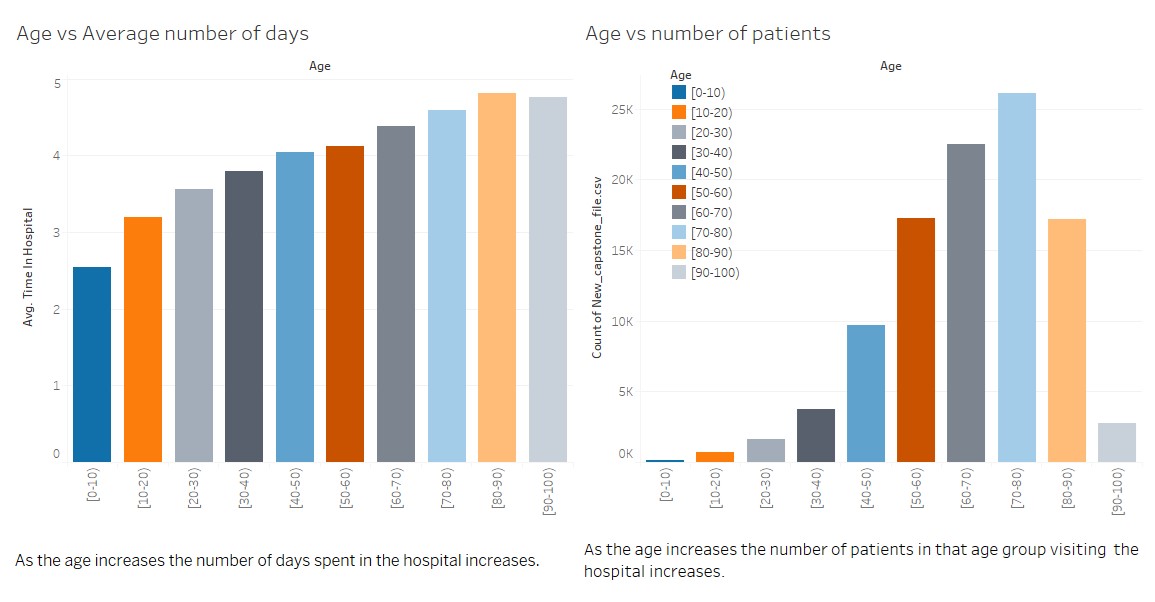
### One-Hot Encoding

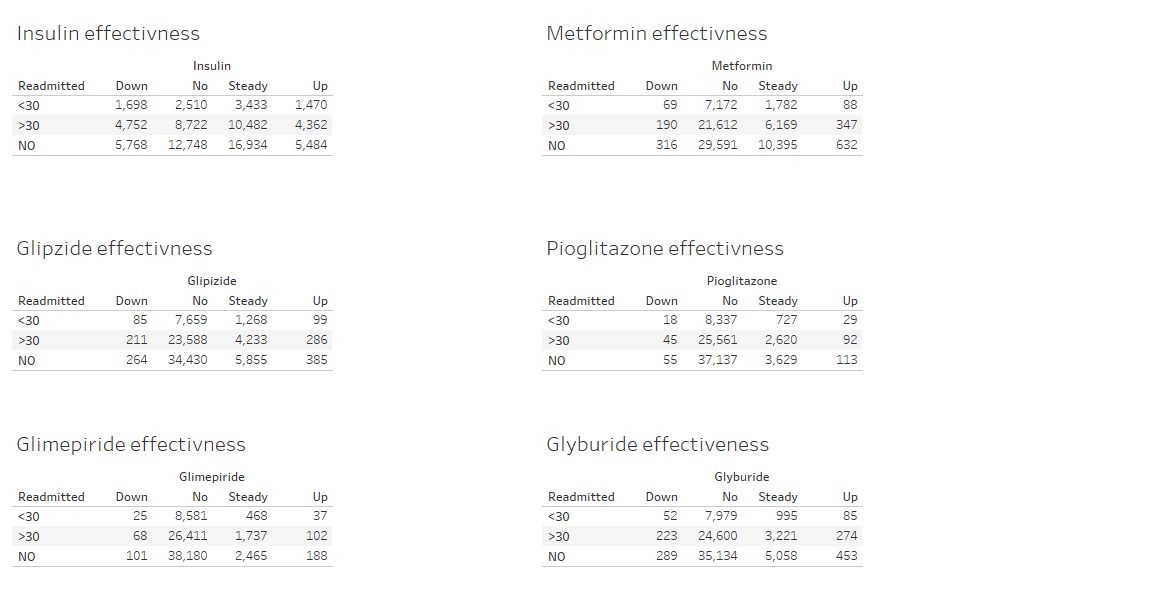
For categorical variables where no ordinal relationship exists, the integer encoding may not be enough, at best, or misleading to the model at worst.

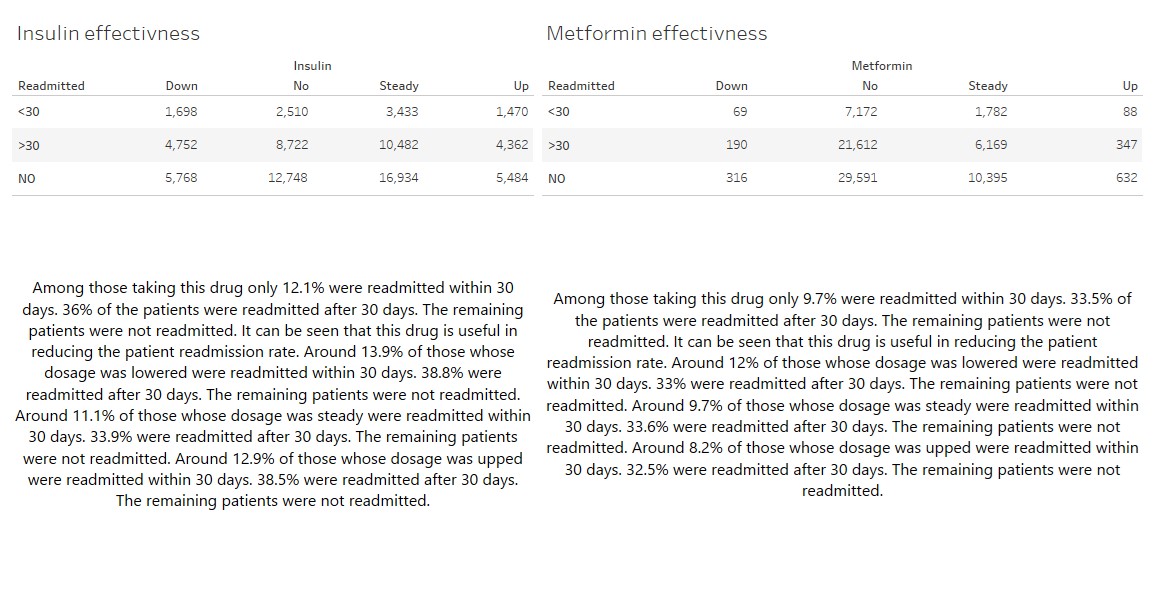
Forcing an ordinal relationship via an ordinal encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

In this case, a one-hot encoding can be applied to the ordinal representation. This is where the integer encoded variable is removed and one new binary variable is added for each unique integer value in the variable.

## Visualizations

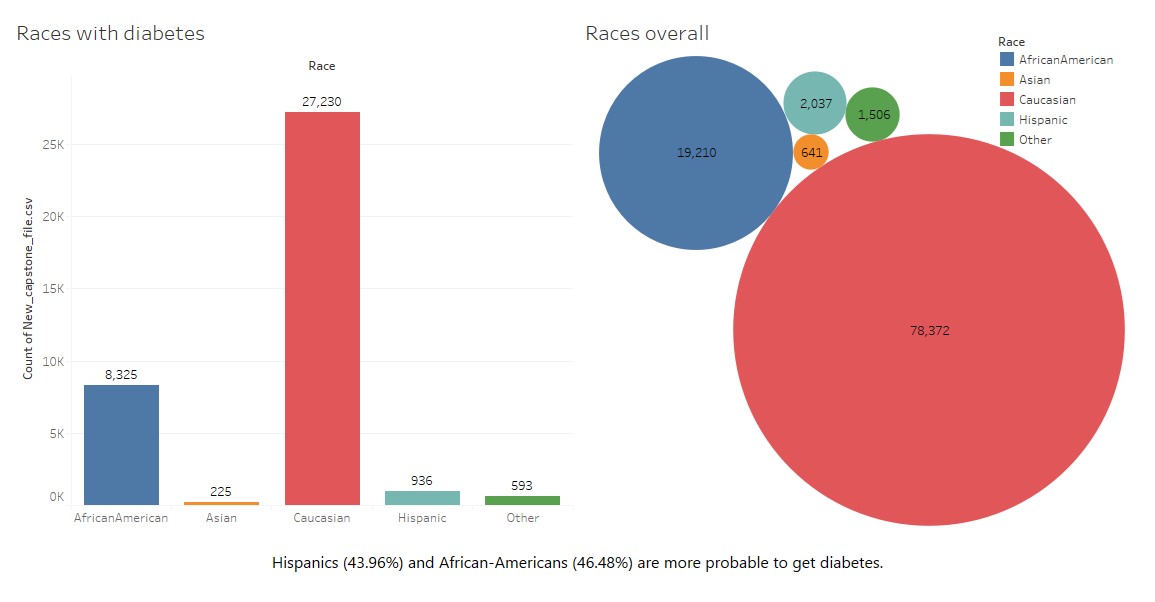






A screenshot of a cell phone

Description automatically generated



A picture containing screenshot

Description automatically generated

A picture containing screenshot

Description automatically generated

A picture containing screenshot

Description automatically generated

**Splitting the dataset into Train and Test**

The dataset is split into Train and Test in the ratio of 70 : 30 using train\_test\_splitter().

## Balancing the Imbalanced dataset

Different balancing techniques (such as Random Oversampling, Random Under sampling, SMOTE were performed. It was observed that SMOTE gives the best performance for the dataset during model building.

# Model Building

Three scenarios were created for model building

* Scenario 1 - There are 3 sub classes NO, >30, <30.
* Scenario 2 – There are 2 Sub classes NO, Readmitted [>30,<30].
* Scenario 3 – There are 2 sub classes >30, <30.

## Models Used

### Logistic Regression

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It’s an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

### Decision Tree Classifier

Decision Tree Classifier is a simple representation for classifying examples where the data is continuously split according to a parameter. It is built through a process of binary recursive partitioning, where an iterative process of splitting the data into partitions and further splitting on each of the branches occurs. Each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric). The leaf nodes of the tree contain an output variable (y) which is used to make a prediction

### Support Vector Classifier

SVC is a non-parametric clustering algorithm that does not make any assumption on the number of shape of the clusters in the data. It works best for low dimensional data. Incase of high dimension data principal component analysis(PCA) is usually required. There are specialized optimization procedures that re-formulate the optimization problem to be a Quadratic Programming problem. The most popular method for fitting SVM is the Sequential Minimal Optimization SMO method that is very efficient. It breaks the problem down into sub-problems that can be solved analytically (by calculating) rather than numerically (by searching or optimizing).

### KNN

KNN is a lazy learning and non-parametric algorithm because it does not have a specialized training phase and it uses all the data for training in classification. It uses feature similarity to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. Because the entire training dataset is stored, one may want to think carefully about the consistency of the training data. It might be a good idea to curate it, update it often as new data becomes available and remove erroneous and outlier data.

### Random Forest Classifier

Random Forest as its name implies, consists of a large number of individual decision trees that operates as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.

Decision trees can suffer from high variance which makes their results fragile to the specific training data used. Building multiple models from samples of your training data, called bagging, can reduce this variance, but the trees are highly correlated. Random Forest is an extension of bagging that in addition to building trees based on multiple samples of your training data, it also constrains the features that can be used to build the trees, forcing trees to be different. This, in turn, can give a lift in performance.

### XGBoost Classifier

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. XGBoost and Gradient Boosting Machines GBMs) are both ensemble tree methods that apply the principle of boosting weak learners CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

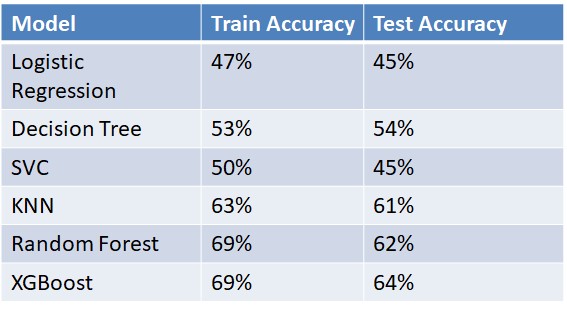
### LightGBM

Light GBM is a gradient boosting framework that uses tree based learning algorithm. Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm. The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as ‘Light’ because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development. Light GBM is sensitive to overfitting and can easily overfit small data.

## Scenario 1

We have 3 subclasses in the target variable. The subclasses are in the ratio of NO : 30 : 30 is to 53% : 35% : 12%. Hence it can be seen that there is class imbalance present in this dataset.

Oversampling was performed to remove this imbalance. As a result the new shape of the dataset is 1,64,000 rows and 26 columns.



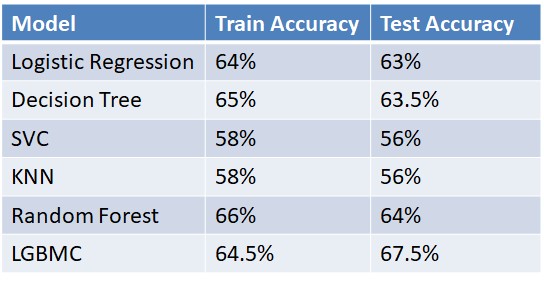
XGBoost has the best accuracy.



## Scenario 2

We have 2 subclasses in the target variable. The subclasses are in the ratio of NO : Readmitted is to

54% : 46%



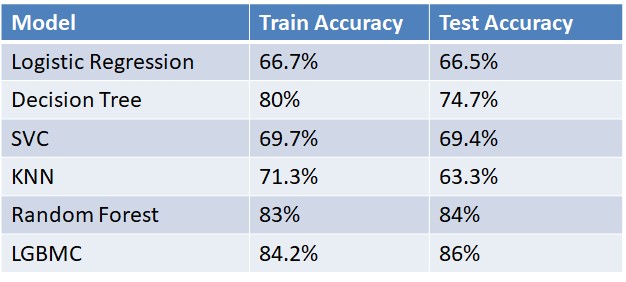
LGBM has the best accuracy.



## Scenario 3

We have 2 subclasses in the target variable. The subclasses are in the ratio of 30 : 30 is to 70% : 30%. Hence it can be seen that there is class imbalance present in this dataset.

Oversampling was performed to remove this imbalance. As a result the new shape of the dataset is 71090 rows and 26 columns.



LGBM has the best accuracy.



# Business Inference

* The Hospital can use our model in scenario 1 to find out whether they should give higher care to a diabetic patient who might get readmitted in the next 30 days to increase their recovery rate.
* The hospital can further use our model in scenario 2 to find out whether a diabetic patient will get readmitted or not to prioritize care.
* If the hospital knows a diabetic patient will get readmitted they can use our model in scenario 3 to find out if the patient will get readmitted before or after 30 days to prioritize care.
* A pharmaceutical company producing metformin can focus on marketing it much more as despite being effective it is not used by a majority of people.
* A pharmaceutical company can target Hispanics and African Americans as they are more probable to get diabetes.
* A pharmaceutical company can target Hispanics for sale of more insulin.