SCHOOL OF INFORMATION STUDIES

SYRACUSE UNIVERSITY

PROJECT REPORT

IST 718 – Big Data Analytics | Guided by Professor Willard Williamson

M001 | Group 2

PREDICTION ON HOSPITAL READMISSION CLASSIFICATION USING DIABETES DATASET

Submitted by:

Akhil Nair | Bhavish Kumar | Laxman Kumar | Saheb Singh

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Diabetes is a widespread chronic disease that is accompanied with irregularities of blood glucose levels due to problems related to insulin. Some of the diabetes patients face a situation of getting readmitted to the hospital due to deterioration of their health condition even after getting discharged, which causes severe inconvenience and distress to the patients. Increase in Hospital readmission rates are an indication of poor hospital quality, poor treatment and they result in the hospital getting penalized. Hence by identifying the factors that lead to higher readmission and being able to predict if a patient is going to be readmitted, the treatment provided by the hospital can be changed, to avoid a readmission and thereby the quality of healthcare provided to the patient can be largely improved, as well as billions of dollars can be saved. For diabetes; the cost analysis estimates that $250 million can be saved across 98,000 diabetic patients by incorporating predictive modeling and prompting greater attention to those who were predicted to get readmitted.

The objective of this project is to help the hospitals accurately predict if a patient is going to get readmitted after discharge. The predictions made will help the hospital make informed decisions on the necessary treatment process alterations in order to reduce patient readmission rate. By accurately predicting if a patient is going to be readmitted, the hospital will be able to make necessary treatment changes to avoid the patient readmission. This reduction in patient readmission rate will help the hospital save billions of dollars and also improve their customer rating.

Prediction on Hospital Readmission Classification using Diabetes dataset is obtained from the UCI ML Repository. The data set represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. The data has around 100k+ observations and 53 attributes with a mix of both categorical and numerical variables.

1. INTRODUCTION

As mentioned above, the goal of the project is to predict if a diabetes patient is going to get readmitted after getting discharged. This prediction will help the hospital mark all those patients who are likely to get readmitted after discharge as 'red' and focus on providing special/altered treatment to such patients who are marked as red. Also, by identifying the factors that are causing a patient to get readmitted, the hospital can take necessary corrective measures towards modifying their treatment which will prevent the patient from getting readmitted. These measures will lead to an enhancement in the treatment quality which will in turn improve the hospital rating and patient satisfaction. This reduction in readmission rates will also help the hospital save a lot of money and avoid lawsuits. Hence there is a strong need to accurately predict if a patient is going to get readmitted after discharge and this is where Data Science techniques come in handy. We will discuss all the data science techniques that have been used to solve this problem in detail, in the upcoming sections.

2. DATASET

The dataset was obtained from the UCI ML repository, as mentioned above. It is a patient level dataset with a total of 101,766 rows representing 101,766 patients along with 53 columns representing 53 different patient attributes for each patient. The dataset contains primary key and foreign key attributes such as 'patient number', 'admission id' etc. The data contains information on the patient admission, discharge and readmission dates. We also have features that capture important information such as number of procedures performed on the patient, total time spent in the hospital, change in medicines, change in insulin levels etc. The data also contains other categorical attributes such as race, admission type, medical specialty of admitting, etc. and binary variables such as gender, Diabetes Medication used and other numeric variables like age, time in hospital, number of lab test performed, number of medication, number of outpatient, inpatient, and emergency visits in the year before the hospitalization etc. All these attributes could potentially have a very high significance in determining the output variable. The target variable has 3 different classes which are "No readmission", "A readmission in less than 30 days" & "A readmission in more than 30 days". But since we are only interested in predicting if the patient is going to get readmitted or not, we will be combining < 30 and > 30 days into one class and call it 'readmitted'.

Below is the table summarizing the statistics of the numeric variables in the dataset. The summary table contains the '*mean'* (to indicate the central tendency), *'standard deviation'* (to indicate the spread) and min-max values (to indicate the range/spread) for the numeric variables present in the dataset. We can see that, there can be a right skew in the 'number outpatient' and the 'number emergency' variables as the max values are far away from the mean value.

summary	num_procedures	num_medications	number_outpatient	number_emergency	number_inpatient
count	101766	101766	101766	101766	101766
mean	1.339730361810428	16.021844230882614	0.36935715268360747	0.19783621248747127	0.635565906098304
stddev	1.7058069791211554	8.127566209167284	1.2672650965326786	0.9304722684224636	1.2628632900973245
min	0	1	0	0	0
max	6	81	42	76	21

Table 1 Descriptive statistics of numeric variables

From the below histogram, we can observe that the both “number of lab procedures” and “number of medications” variables are almost normally distributed for both readmitted and not readmitted classes.

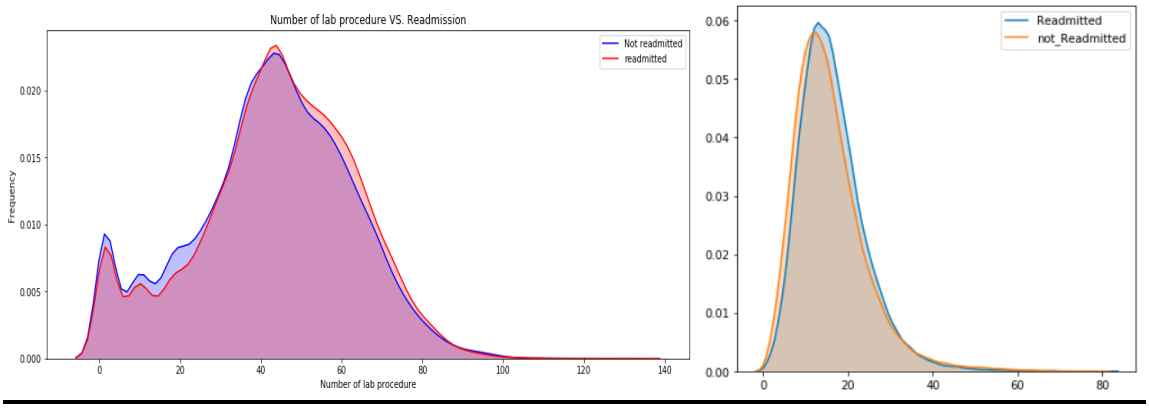


Figure 1 Histogram of number of lab procedures and number of medications

Next by looking at the frequency distribution of our target variable, “readmitted”, shown in the below table, we can observe that the data is mostly balanced with only a difference of around 7K rows between the class 0 and class 1.

readmitted	count
0	54864
1	46902

Table 2 Frequency distribution of target variable ‘readmitted’

Next, we will discuss the methodology we followed to complete this project. We have employed the CRISP-DM iterative process, which is the Cross Industry Standard Process of Data Mining. The process starts off with Data Cleaning to handle the data quality issues, followed by Data Preprocessing, Exploratory Data Analysis and feature selection, which is followed by model building, model performance evaluation, fine tuning the model and producing the results, inferences and conclusion. We have built binary classification machine learning models to accomplish our project objective of predicting whether a patient is going to get readmitted or not. These 7 different models are built using 7 different supervised machine learning algorithms. The presence of a target variable is what makes it supervised learning.

We will discuss each stage of the process and all of the supervised learning algorithms that we have used in detail one by one in the upcoming sections.

3. METHODOLOGY

In this section we will cover every stage of the Data Science methodology that we have used to accomplish our project objective.

3.1 Data Preprocessing

The first step of the Data Preprocessing stage is to perform data wrangling, which is also known as Data Cleaning. **Data wrangling** is the process of cleaning, structuring and enriching raw data into a desired format for better decision making in less time and it is done to handle all the data quality issues that exist in the data. Some examples of the data quality issues that people often encounter when working on data science projects are presence of duplicate values, missing values, noise/outliers, attributes with low to no variance etc. Let us discuss the data quality issues that we encountered and how we handled them.

Firstly, we witnessed the presence of special characters and invalid values in some of the attributes in the dataset. We declared all such values as missing values and set it as Null for future treatment. Then we dropped all the attributes which had more than 50 % of missing values because of which the columns 'weight', 'payer code' & 'medical speciality' got dropped.

Secondly, we identified 7 columns ('race', 'diag_1' ,'diag_2', 'diag_3', 'admission_type_name', 'dischage_disposition_name', 'admission_source_name') which had only a few missing values. Since all these 7 attributes are categorical, we imputed the missing values with the Mode (most frequently occuring value) of that attribute, which helped us to reduce the number of missing values in the dataset to zero.

Next, we identified all the primary key and foreign key attributes such as ‘patient number’, ‘admission type id’ etc. which have no role in explaining our target variable and are insignificant. Hence such attributes were dropped from the dataset.

Next, we identified all the attributes that have either No Variance or close to no variance, i.e. all the attributes where the majority of rows contain the exact same value. For example if we look at the distribution of the ‘troglitazone’ variable, we can see that the majority of the values belong to the “No” class and only 3 values are from the “Steady” class. Such attributes with No variance are of very little use in explaining our target variable and hence such attributes can be dropped.

troglitazone	count
Steady	3
No	101763

Table 3 Troglitazone count

We identified 10 such attributes (‘'acetohexamide', 'tolbutamide', 'troglitazone', 'tolazamide', 'examide', 'citoglipton', 'glipizide-metformin', 'glimepiride-pioglitazone', 'metformin-rosiglitazone'& 'metformin-pioglitazone') and dropped them.

Lastly, we plotted histograms of all the numeric variables to detect the presence of outliers in the data by looking at their distributions. We did not find a sufficiently strong reason to declare any value as outlier and hence we did not perform any outlier treatment such as trimming or winsorizing as there were no outliers in our dataset.

After we completed Data Wrangling, we applied some data transformation techniques such as **One Hot Encoding** and **Standardization** as part of Data Preprocessing. One Hot Encoding is the process of

converting all categorical variables into numeric dummy variables. One Hot encoding is necessary since all our data has to be numeric in order to build machine learning algorithms. Standardization is done prior to Principal Component Analysis since the variables have different units and values in completely different ranges. Standardization transforms every variable to new values which are known as Z score values. These new Z score values will have a mean of 0 and standard deviation of 1. Setting the standard deviation to 1 is optional and it depends on whether or not the values have the same units. In our case, since all the values have different units, we decided to set the standard deviation also to 1. Hence the Z score values obtained after standardization are calculated as z = (Xi - Mean)/standard deviation. Now let us discuss Principal Component Analysis and the purpose behind doing PCA and its outcome.

3.2 Principal Component Analysis

PCA is generally done to identify the importance of features for variable selection and also for dimensionality reduction. We have performed PCA because we were interested in identifying the most important attributes that have a major role in predicting if a patient is going to get readmitted or not. Since PCA requires all the attributes to be numeric with mean = 0, One Hot Encoding followed by standard scaling was done on the dataset as mentioned above. On this processed data 2 principal components were built to identify the most important features that explain the maximum variation in the data. Since the first 2 Principal Components explain most of the variation in the data, features that have the highest absolute value of loading vector coefficients in the first 2 Principal Components can be considered as the most important features in the dataset. In the below shown figure, we can see the 10 most important features from Principal Component1 and Principal Component2, that explain the maximum variance in the data.

	feature	abs_loading
2378	changeNo	0.383429
2379	changeCh	0.383429
2381	diabetesMedNo	0.361159
2380	diabetesMedYes	0.361159
2370	insulinNo	0.277394
2326	metforminNo	0.188435
2385	num_medications	0.182882
2327	metforminSteady	0.171727
2373	insulinUp	0.144420
2372	insulinDown	0.141952

	feature	abs_loading
2258	admission_type_name_Emergency	0.368400
2289	admission_source_name Emergency Room	0.367960
2290	admission_source_name Physician Referral	0.360340
2259	admission_type_name_Elective	0.322756
2383	num_lab_procedures	0.218369
2384	num_procedures	0.155462
2389	number_diagnoses	0.139227
6	diag_1_414	0.124082
12	diag_1_715	0.121701
2260	admission_type_name_Urgent	0.111406

Figure 2 Most important features with highest loading vector coefficients

We then plotted the cumulative Explained Variance plot and Scree Plot as shown in the figure below to identify how much of variance in the data is explained by how many principal components. We can see that approximately 2000 Principal Components are required to explain around 80% of the variance in the data which is because of the sparse nature of our one hot encoded data. Since our dataset had a lot of categorical variables of which a few variables have around 700 levels, because of which our one hot

encoded dataset was very sparse with 2390 columns. Because of this sparse nature of our dataset, the first 2 principal components could only explain around 0.5 % of the variance in the data and also as many as 2000 principal components are required to explain around 80% of the variance in the data.

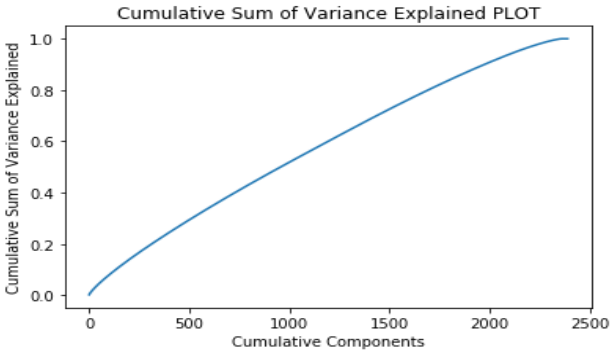


Figure 3 Cumulative Plot of Explained Variance

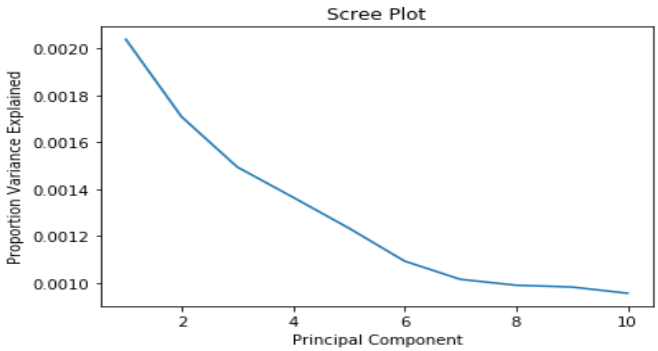


Figure 4 Scree Plot Explained Variance

From these two plots we can understand that PCA is not performing up to the mark on our dataset, as ideally the first 2 principal components are supposed to explain a large proportion of the variance in the dataset. Because of this lack of variance explained by the first few principal components, we felt that selecting a reduced number of features for model building based on our PCA results may not be a very good idea and hence we decided to use other methods also to identify the most important features. We obtained the variable importance output of our Random Forest and Gradient Boosted Tree models also to identify the most important variables in predicting our target variable. We will look at these other methods in more detail in the upcoming sections.

3.3 Exploratory Data Analysis

Exploratory Data Analysis is the process where we search for relationships in data or any hidden insight. These relationships will help us further while model generation. During the exploratory data analysis process we make various univariate, bivariate and multivariate plots in order to find the interesting pattern in the dataset. All the patterns are shown as story by following the principles of explanatory data analysis.

The figure 5 below gives the information about the number of cases grouped by race. About 77% of the total cases are from 'Caucasian' race followed by 'African American' with around 18% of total cases. Only 46% of total 'Caucasian' cases were readmitted and the rest 53.5% were not readmitted. Of all the 'African American' cases, 45.7% patients were readmitted to the hospital. Both the Caucasian and African American are the major races in the dataset, about 95% of the total data belongs to these two races. 2.32% of the total data belongs to 'Hispanic' followed by 'Others' with 1.47%. Asian with the least number of cases with only 0.62% total cases.

Figure 6 gives the information about the count of cases by gender and whether they are admitted again or not. The dataset is balanced w.r.t. Gender since we have around 53% Females nad 47% males. Among the female cases, around 47% cases were readmitted again on the other hand 45% of total male cases were readmitted again.

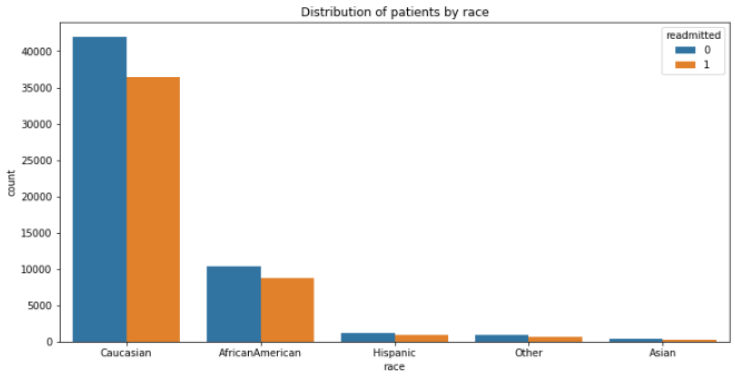


Figure 5 Number of cases grouped by Race

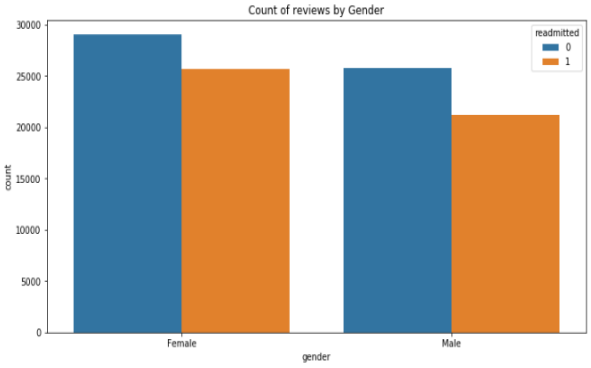


Figure 6 Count of cases by Gender

Figure 7 below is a bar plot of age-group grouped w.r.t to readmission and gender, data is left-skewed with people within the age group of 70-80 having the highest number of counts in the dataset. The reason can be many and one of the reasons can be since the average life span of human life in the USA is approximately 78 according to CDC (link). Number of female patients in comparison with male after 70 years of age is higher with great margin. People within age group 60-70 are the second most number of patients admitted to the hospital followed by age group 50-60 and 80-90.

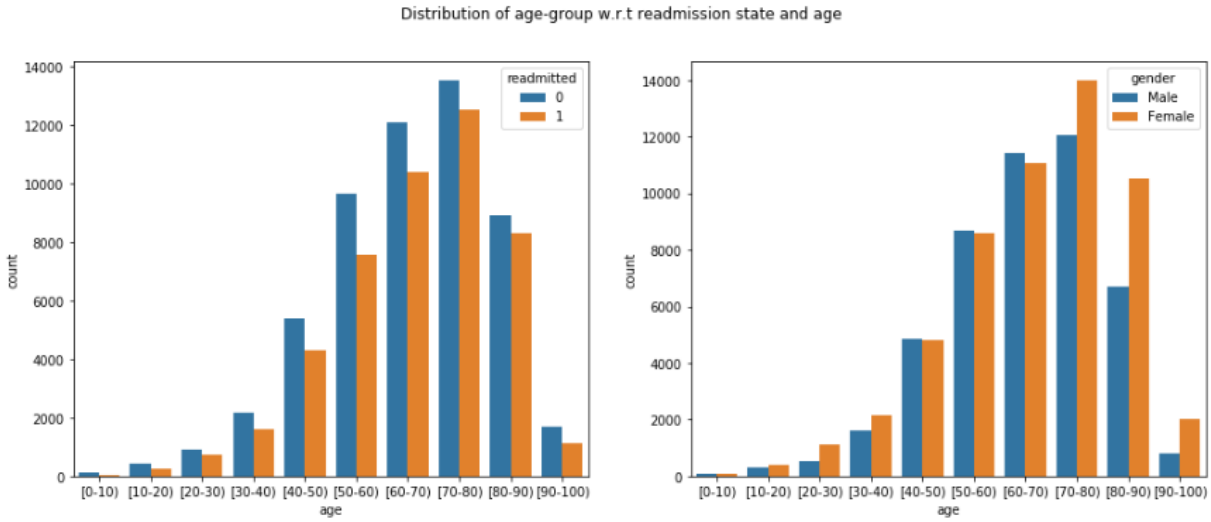


Figure 7 Distribution of age-group w.r.t readmission rate gender

Figure 8 and Figure 9 gives us the distribution about the diabetes med and insulin. Figure 8 is a bar plot describing whether the patients are on diabetes medicine or not. And what is the distribution among these 2 groups about the readmission. From figure 8, we can infer that we have a higher number of patients who are on diabetes medicine. About 77% of the patients take diabetes med and among those about 52% have not readmitted again. But in the case where people do not take diabetes med, about 59% of the total have not readmitted again.

Figure 9 gives the information about how the insulin is distributed and whether it affects the readmission of patients. More than 46% of the patients are not given insulin and among those 56% of the patients are not readmitted again. While the patients who are given insulin at a steady rate are about 30% of the data

and 54% of those were not readmitted again. In these both cases we have higher number of the patients who were not readmitted. But when the insulin doses were up and down we found that chances of readmission were high.

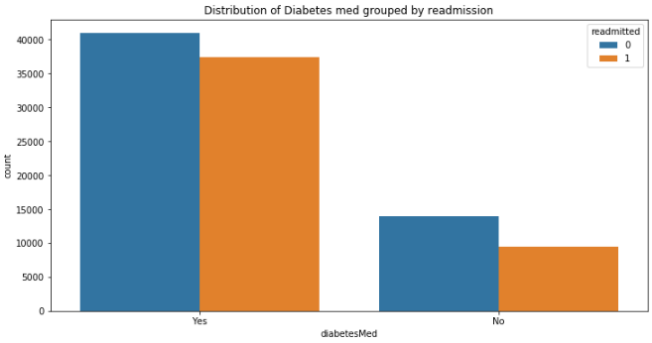


Figure 8 Diabetes med distribution

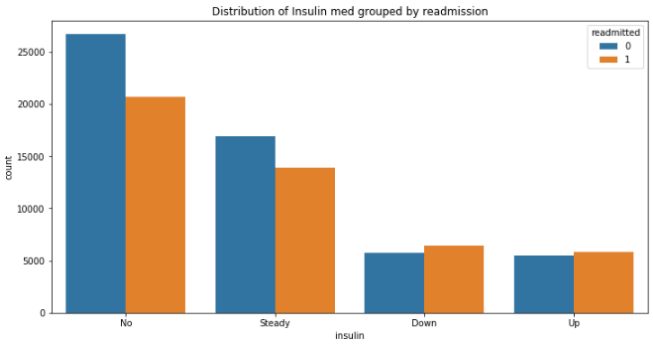


Figure 9 Distribution of Insulin

Figure 10 is a kernel density plot of time in the hospital column for each of the readmission states. People who were not readmitted again have higher frequency in the initial 4 days. But as the number of days in hospital increases we can notice that frequency of people readmitting again in the hospital increases.

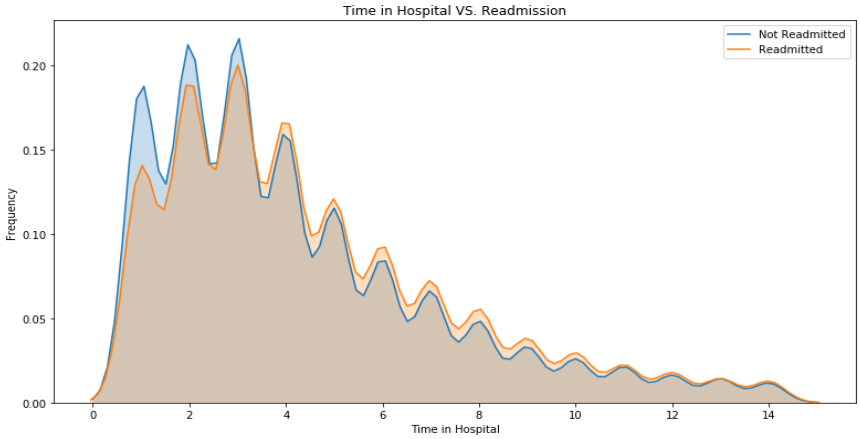


Figure 10 Distribution of Time in Hospital

4. EXPERIMENT

Now coming to the model building and evaluation process. Since ours is a binary classification problem, we have built seven classification models to predict our binary target variable (patient readmission). The seven classification algorithms which we have used for building our models are:

- Logistic Regression
- Decision Tree
- Random Forest
- Gradient Boosted Trees
- Naive Bayes Classifier

- Support Vector Machine (Linear)
- Multi Layer Perceptron

We will discuss each one of these models in detail in the upcoming sections.

We have used **Accuracy**, **Recall** and **AUC** as our main model performance evaluation metrics. Since our dataset is balanced Accuracy is certainly a valid choice for evaluating our model performance. Recall is also another important metric in our case because it is important for us to minimize False Negatives. False Negatives are cases where the patient is going to get readmitted, but our model prediction says that the patient is not going to get readmitted. Such cases are highly undesirable and must be minimized by focusing on improving our Recall scores. The third most important metric for us is AUC (Area Under ROC), which indicates how well the predicted probabilities from the positive classes are separated from the negative classes. Now let us look at each of these 7 models one by one.

4.1 Logistic Regression

Logistic regression is a 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 classes. Logistic regression is an excellent model to use when the features are linearly separable. One advantage of logistic regression is the model is interpretable — i.e. we know which features are important for predicting positive or negative.

With the starting assumption that the impact of factors and their interactions can be modeled as a log likelihood of outcome, logistic regression helped us understand the relative impact and statistical significance of each factor on the probability of readmission. We handled missing values by either removing the columns or following the preprocessing steps. We tried both L1 and L2 regularization to penalize complex models. We also tried Randomized Logistic Regression to get the most important features for our logistic regression model but got poor performance metrics than the feature sets considered above and hence, we decided to stick with the feature sets above to maintain consistency.

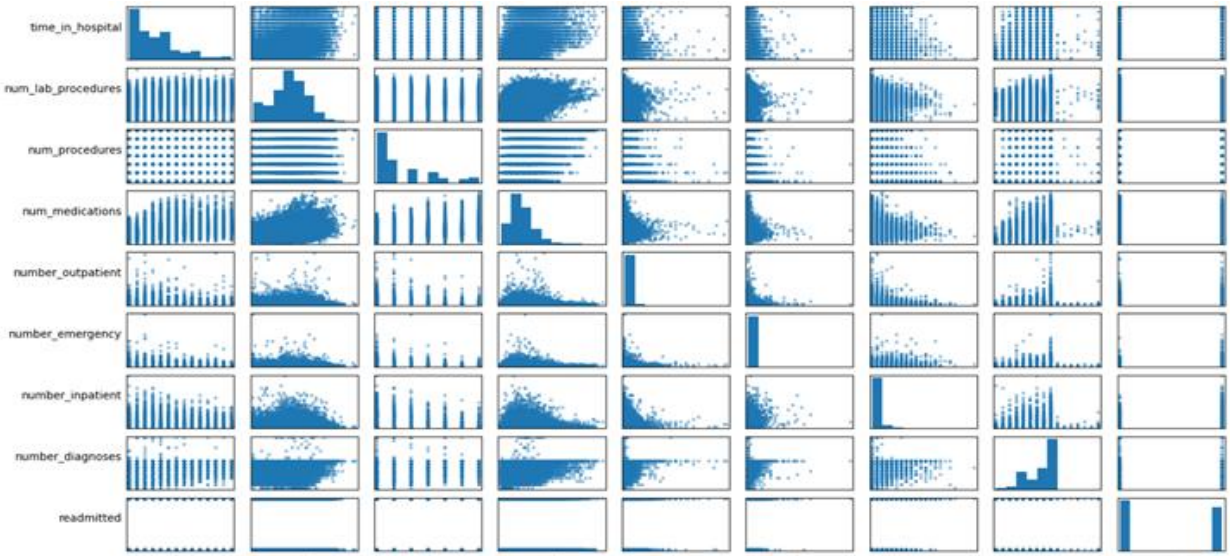


Figure 11 Scatter plot of numerical features

From the pairwise scatter plot we understood the distribution of data to make better hypotheses to include the features in our model

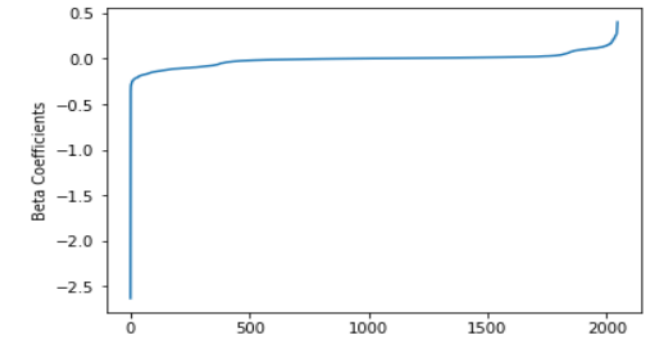


Figure 12 Beta Coefficient plot

We also observed how beta coefficients change during modelling phase to understand which features are impactful to make a better prediction of the target variable.

We used GridSearchCV function to perform an exhaustive search for the best C in just one line of code. The process also uses cross validation and an l2 penalty, which both help manage variance and in turn keep the model from becoming too general to the point that its predictive value is weakened.

Precision reflects the percentage of positive (i.e., readmitted within 30 days) predictions that are correct. If the model classifies 100 observations as positive, and 80 of those were in fact positive, then the model's precision is .8 (or 80%).Recall captures the percent of true positives that are classified as positive. Returning to our model's context, if there were 1,000 patients readmitted within thirty days and the model detected 850 of them, then the model's recall would be .85 (or 85%).Finally, the F1-score (the harmonic mean) shows the weighted average of precision and recall. It is a useful metric because it factors in both false positives and false negatives. The model here has an F1-score of .85, which is pretty good considering that 1.00 would be a perfect score and .5 would be as good as random guessing. However, this higher score masks the model's weak performance with positive cases. The final model performed with an accuracy of 60.10 %, a recall of 53.96 % and an AUC score of 59.64 %.

4.2 Decision Tree

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.By iteratively and hierarchically observing the level of certainty of predicting whether someone would be readmitted or not, we find the relative importance of different factors using a more human-like decision making strategy in establishing this determination.

We can see much improvement in all metrics compared to logistic regression. We explored several model parameters for fine-tuning such as depth of tree, minimum sample size and type of information gain function (gini vs entropy) etc. =Grid search was effective to find the optimal parameters for best performance (assuming the computation isn't taking too much time). We also tuned other hyperparameters

for eg. we used gini impurity to make a decision at the node split. While the max depth of the decision tree is 5 and total number of nodes is 29

But that's not all we can get from decision trees. Each node of the tree indicates a data splitting decision that gets us closer and closer to predicting readmission. The challenge is visualizing deep trees with lots of nodes. So for sake of simplicity, let's visualize only the first two levels of this decision tree.

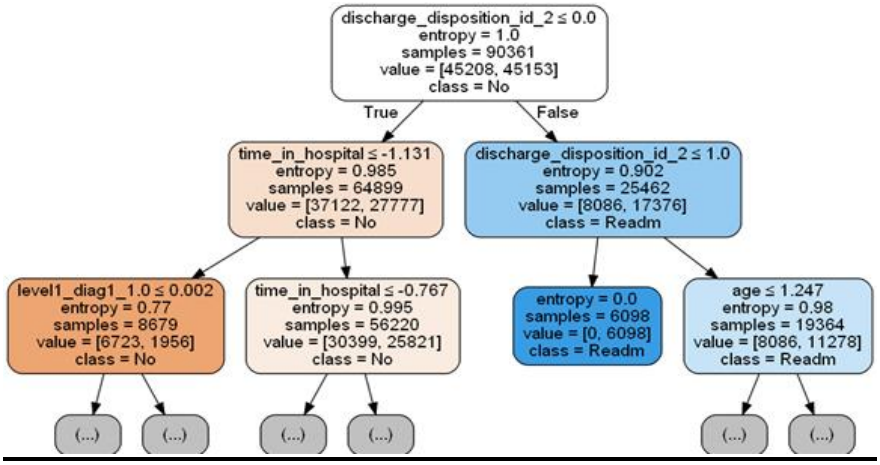


Figure 13 Decision Tree plot

Looking at this, we can tell that the first feature used in deciding whether a patient will get readmitted or not, is whether the patient is discharged to another hospital or facility (discharge_disposition_id_2… category). Next level of the tree shows this same feature repeated on one hand and days spent in hospital on the other. And so no. But this can easily get confusing, so we would like to summarize this information in some way.

Recall is a (percentage) measure of how many cases were identified correctly -- or detected -- in a given (positive or negative) class. It is also know as the true-positive or true-negative rate. In this case, for patients who were in fact readmitted within 30 days, the recall is 40.95%.While the accuracy of the model was 56.89% and an AUC score of 55.69%.

4.3 Random Forest

Random Forest is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. After applying Random Forest, we get an accuracy of regardless of using Gini or Entropy function. Feature importance are only slightly different using these two functions, and we show the one with Gini function below.

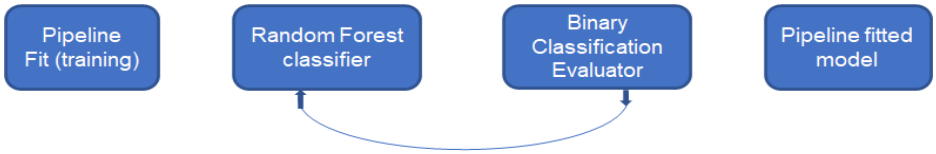


Figure 14 Pipeline of Random Forest

Our decision tree model indicates highest importance of admission source name, time spent in the hospital, number of medications and gender for both simple and complex versions. Since we used cross-validation and achieved similar test and train accuracy to avoid overfitting, this may suggest a different correlation structure of the variables in the model or lower explained variance in logistic model. For random forest ensemble, almost similar features had high importance, although the distribution was more even as compared to decision tree. This is likely due to stabilization of importance across many trees. Recall is a (percentage) measure of how many cases were identified correctly -- or detected -- in a given (positive or negative) class. It is also known as the true-positive or true-negative rate. In this case, for patients who were in fact readmitted within 30 days, the recall is 48.51%. While the accuracy of the model was 54.12% and an AUC score of 50.12%.

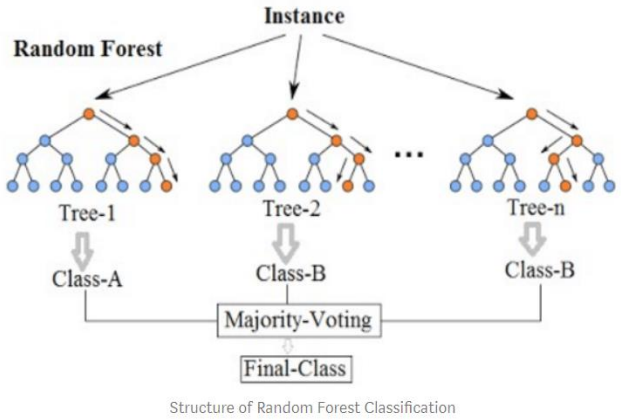


Figure 15 Random Forest

	feature	importance
0	admission_source_name	0.035383
1	time_in_hospital	0.013409
2	num_medications	0.012223
3	glimepiride	0.011492
4	A1Cresult	0.006636
5	gender	0.005338
6	number_outpatient	0.003724
7	diag_1	0.002749
8	glyburide-metformin	0.002134
9	rosiglitazone	0.002121

Figure 16 Top 10 important features

4.4 Gradient Boosted Tree

Gradient Boosted Tree is an ensemble learning method which follows the wisdom of the crowd principle to make classification decisions by considering the output of several decision trees that are built sequentially. The final model is produced by taking the weighted average of predictions made by each base classifier. The GBM also uses the splitting criteria as Gini Index or Entropy similar to a decision tree, as GBM is nothing but multiple decision trees built sequentially. Each tree is grown using information from previously grown trees. GBM works iteratively as mentioned above to reduce the loss function and by updating the residuals after every iteration the log loss will be reduced. We started off by building the baseline Gradient Boosting Model without tuning any of the hyperparameters. After evaluating the performance of the baseline model on the validation dataset, we tuned the hyperparameters to try and improve the model performance.

From the Gradient Boosting model we obtained the variable importance as shown in the below figure, to identify the most important variables that play a major role in building the model. We did not select a limited number of features for building the subsequent models based on the variable importance results

obtained from the Gradient Boosting Tree model because our model was performing well when we included all the 34 predictor variables and hence we chose not to drop any variable. After building the baseline model, we built a tuned model by performing a grid search with 3 fold cross validation and tuned the hyperparameters feature Subset Strategy, stepSize, maxDepth, and maxIterations to produce a tuned GBT model. The *feature Subset Strategy* is used to pick a strategy for selecting a number of features to be considered for the best split to make. By selecting a limited number of features to be considered for splitting, the correlation between the trees can be reduced. The *stepSize* hyperparameter is to control the learning rate of the model. The stepSize is used to control for overfitting by reducing the model variance which can be done by reducing the learning rate value. The *maxDepth* parameter is again used to reduce the model variance by pruning the tree and preventing it from over growing. The hyperparameter *maxIterations* refers to the number of epochs/iterations the gradient boosting model must undergo in order to reduce the model bias. To control overfitting increase the $number\ of\ trees$ in the model, increase the $number\ of\ features$ required in a node and lower $learning_rate$for the model.

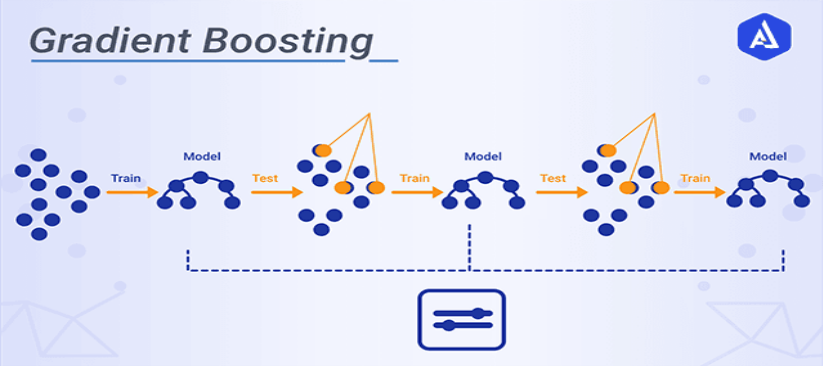


Figure 17 Pictorial representation of Gradient Boosting Tree algorithm

The tuned model was evaluated using model performance evaluation metrics. This model evaluation was done on the validation dataset which the model has not seen before, as the model was trained on the training dataset. The training and validation data split is done randomly by following a 60:40 or a 70:30 split. The process of splitting the dataset into train and validation is known as the hold one out method of model evaluation. This is important because we need to ensure that the model performs well not only on the training data but also on the validation data which the model has not seen before. By following this procedure, we can be assured that the problem of overfitting does not exist in our model and the model variance/flexibility can be controlled without having to compromise by having a low bias. In our case the model performance was in line on both the training and validation dataset and we got a validation accuracy of 64%.

We also produced the variable importance table from the GBT model as shown below. This table shows us the importance of each of the variables towards building the best performing GBT model. The 10 most important variables as per our GBT model is shown in the figure here.

This feature importance was obtained from the Gradient Boosting Model, for the purpose of deriving some inference from the model. From this we can infer that the features ‘Race’, ‘Discharge Disposition Name’ and ‘metformin’ are the 3 most important features for building the GBT model.

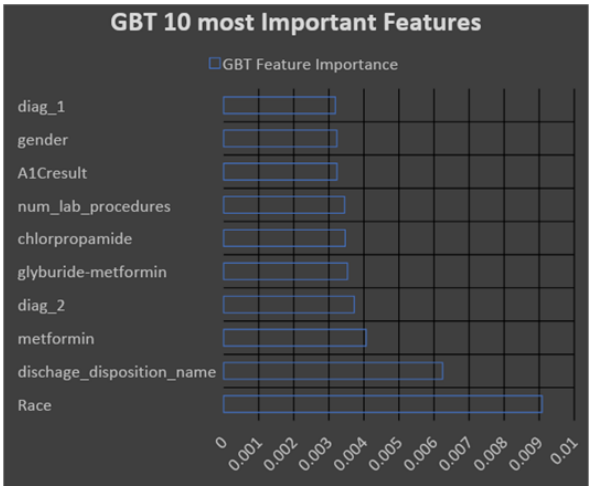


Figure 18 10 most important features obtained from GBT

4.5 Naive Bayes Classifier

Naïve Bayes is from the family of Probabilistic classifiers. Naive Bayes is based upon Bayes theorem. A Naive Bayes model multiplies several different calculated probabilities together to identify the probability that something is true, or false. Naive Bayes is highly scalable and can handle both continuous and discrete values which makes it suitable for tasks involving such a high amount of data with so many different types of features.

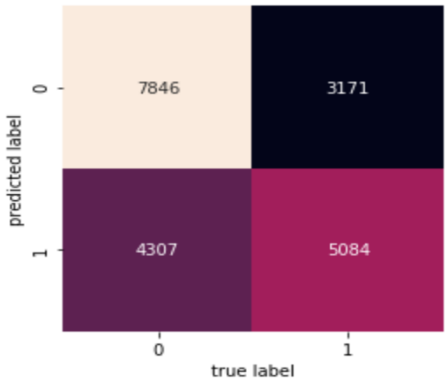
Feature selection is an important task intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable. For feature selection I started removing very less or negatively correlated with our dependent variable. I decided to try scaling to see if it results in improving the accuracy. I wrote a function to check every quantitative feature and check and remove every feature with kurtosis between -2 and 2. After that I used a standard scaler to scale every remaining feature. After selecting features I divided the dataset into training and testing set, we have used 80:20 train and test ratio. Since we have a large dataset 20% data for testing is enough. We got approx. 20,000 rows for testing.

```
# All catgorical features
cat_features = list(df.toPandas().select_dtypes(include=['object']).columns)
# All numerical features
num_features = ['time_in_hospital','num_lab_procedures','num_procedures','num_medications',
                'number_outpatient','number_inpatient','number_diagnoses']
```

Figure 19 Selected feature for Naive Bayes Classifier

From the classification report and confusion matrix in figure, we can see that we have high precision for class 0 by 3% and we also have a high recall for class 0. Class 0 has a high recall as we have very less false negative for class 0 which means that classifier was able to predict class 0 when data point was from class 0 and predicted class 1 a few times. Class 0 was correctly predicted 7,846 times and wrongly predicted as class 1 3,171 times. Whereas for class 1 it was correctly predicted around 5000 times but was

wrongly predicted as class 0 4,307 times therefore it did not perform well while predicting class 1 but got an overall accuracy of 63% and AUC score of 44%.



	precision	recall	f1-score	support
0.0	0.65	0.71	0.68	11017
1.0	0.62	0.54	0.58	9391
accuracy			0.63	20408
macro avg	0.63	0.63	0.63	20408
weighted avg	0.63	0.63	0.63	20408

Figure 20 Confusion Matrix of Naive Bayes

Figure 21 Classification Metrics report for Naive Bayes

4.6 Linear SVM

Linear SVM is a supervised learning regression and classifier algorithm. It is a part of the SVM with the kernel as 'Linear'. Linear SVM is an extremely fast algorithm for solving multiclass classification problems from a very large dataset. It is a linearly scalable routine which creates a SVM model in CPU time which scales linearly with size of training data.

For feature selection, firstly I started removing features which are very less or negatively correlated with our dependent variable. After reaching a threshold, where removing any of the features results in lowering the accuracy I got a set of 15 features mentioned in the figure 22. All the features finalized have either higher correlation or have better bivariate plot with readmission. For the hyperparameter tuning, I used a grid builder at first and replaced the model with the best model's parameter. Only two of the hyperparameter values are different from their default value. Finally I got maxIter as 1500 and regParam as 0.01. We have splitted the dataset into 90:10 train and set ratio. Since we have a very large dataset 10% data for testing is sufficient. We got approx 10,000 rows for the testing

```
#List of all categorical feature columns
categorical_columns= ['race','diag_1','diag_2','diag_3','admission_type_name','dischage_disposition_name',
                      'admission_source_name','gender','age','max_glu_serum','insulin','change','diabetesMed']
#List of all numerical feature column
numerical_columns = [ 'num_lab_procedures','num_medications','number_outpatient','number_emergency',
                      'number_inpatient','number_diagnoses']
```

Figure 22 Selected Feature for Linear SVM

From the classification report and confusion matrix generated using sklearn in figure 23 and figure 24 respectively. We have high precision for class 1 by 1% but we have very high recall for class 0. We got a high recall for class 0 since we have very less false negative for class 0 which means the classifier was successful in predicting the class 0 when it was actually belonging to class 0 and it has predicted class 0 as class 1 very few times. We got 8441 correct predictions for class 0 out of total 10874 class 0 labels. On the other hand we have very less recall for class 1 since only 4403 were correctly predicted as class 1 out

of 9428. Less than 50% of the labels were correctly predicted as class 1 when they actually belong to class 1. Overall we got 63% accuracy and 68 AUC score.

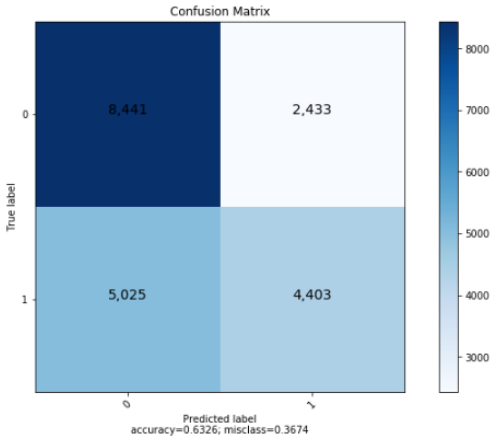


Figure 23 Confusion Matrix of SVM

	precision	recall	f1-score	support
0	0.63	0.78	0.69	10874
1	0.64	0.47	0.54	9428
accuracy			0.63	20302
macro avg	0.64	0.62	0.62	20302
weighted avg	0.63	0.63	0.62	20302

Figure 24 Classification Metric of SVM

4.7 Multi-Layer Perceptron

Multilayer perceptron classifier is based on a feed forward neural network. It contains an input layer and a set of hidden layers. All the hidden layers are fully connected with each other. Nodes in the intermediate layer use sigmoid function while nodes at the output layer use softmax function.

Feature engineering is a very important task for any model preparation. Initially I had fed the network with all the available features and recorded the accuracy. In the second step, I started removing numerical features one by one based on the correlation matrix generated. For the categorical features, I began removing some of the less correlated features based on the jitter plot. At the end of this process, I got 14 features as seen in figure 25- 11 categorical and 4 numerical out of total 34 features. Hyperparameter tuning affects accuracy in a serious manner. Tuning the hyperparameter in an appropriate way is very important. I had manually tried a few combinations of hyperparameters and after running 15 different combinations of maxIter, stepSize and blockwise I was able to finalize the maxIter as 100, stepSize 0.05 and blockSize as 64. The final model contains one input layer with (1x 1541) dimension as input. Figure 26 shows dimension of one row sparse vector. The dimension of the input layer is the size of the sparse matrix of a row in the dataset. After the input layer, we have one hidden layer with 512 neurons and finally an output layer with 2 neurons since our problem statement is a binary classification problem of readmission state false and true. We have used a holdout method for building the model. We have splitted the dataset into a 70:30 train:test ratio. We got approx 30,000 rows for testing which is a very large set but due to computation limitation it was difficult to increase the train set.

```
#List of all categorical feature columns
categorical_columns= ['race','diag_1','diag_2','admission_type_name','dischage_disposition_name',
                      'admission_source_name','gender','age','insulin','change','diabetesMed']
#List of all numerical feature column
numerical_columns = [ 'num_medications','number_emergency','number_inpatient','number_diagnoses']
```

Figure 25 Selected features for perceptron model

```
SparseVector(1541, {0: 1.0, 5: 1.0, 721: 1.0, 1469: 1.0, 1476: 1.0, 1500: 1.0, 1517: 1.0, 1519: 1.0, 1530: 1.0, 1533: 1.0, 153
5: 1.0, 1537: 21.0, 1539: 1.0, 1540: 9.0})
```

Figure 26 Row of data as Sparse Vector

From classification reports in figure 27 and confusion matrix in figure 28, we can observe that we got accuracy of 64% and AUC as 64%. We have higher precision and recall for class 0 then class 1. Higher recall shows that the classifier was able to correctly detect class '0' and there are very less false negatives. Also high precision for class 0 indicates that classifier has correctly detected class 0 as class 0 and very few cases where the class is actually 1 and predicted as class 0. Out of 16511 class 0 labels only 12022 were actually predicted as class 0 and rest 4489 were predicted as class 1. On the other hand out of 14091 class 1 labels, only 7585 were actually predicted as class 1 and rest 6505 were predicted as class 0. It can be seen that the classifier finds it difficult to predict the class 1 in comparison with class 0 and therefore we have very low recall for class 1.

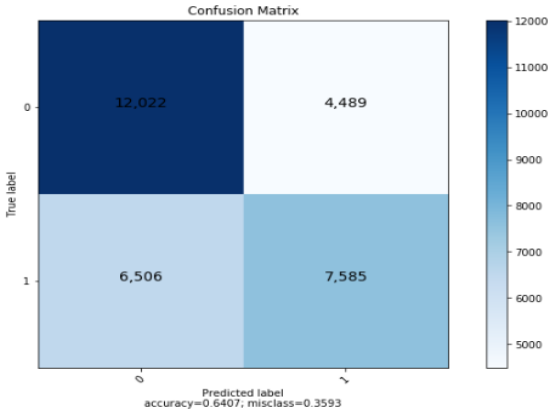


Figure 27 Confusion Matrix of MLP

	precision	recall	f1-score	support
0	0.65	0.73	0.69	16511
1	0.63	0.54	0.58	14091
accuracy			0.64	30602
macro avg	0.64	0.63	0.63	30602
weighted avg	0.64	0.64	0.64	30602

Figure 28 Classification metric report of MLP

5. RESULT

For our hospital readmission dataset, Gradient boosting tree and Naïve Bayes performed best with an accuracy of 64 % and 62 %. While due to sparse nature of our dataset tree-based models i.e. Decision Tree and Random Forest gave an accuracy of 56 % and 54 % respectively. While neural nets-based algorithm Multi-Layer Perceptron had an AUC score of 60.5 %. Gradient Boosted tree had the best AUC score of 69%

ALGORITHM	ACCURACY	RECALL	AUC
Logistic Regression	60%	53%	59%
Decision Tree	56%	41%	55%
Random Forest	54%	48%	50%
Gradient Boosted Tree	64%	52%	69%
Naive Bayes Classifier	63%	54%	44%
Linear SVM	64%	63%	68%
Multi-layer Perceptron	64%	63%	64%

6. CONCLUSION

From the project results we can conclude that the ensemble learning methods and the deep learning methods are the ones that usually produce the best performing models, which is clearly evident frosm the results table shown above. We can also conclude that hyperparameter tuning through grid search and K fold cross validation are very much essential when building a machine learning model where minimal bias and variance are desired. This was proven to be true with all our models where the model performance improved upon performing hyperparameter tuning and K fold cross validation. Another conclusion that we can draw is that PCA is not a good method to perform feature selection or do dimensionality reduction, when dealing with high dimensional sparse data, as a very large number of principal components will be required to explain a satisfactory amount of variance in the data.

Through this project we got hands-on experience of building machine learning models by using pipelines, estimators and transformers available in the PySpark libraries. We got experience of working on datasets which are very big in size and contain a large number of rows and columns. This project also taught us the importance of using github for version control and collaboration.

The goal of this project was to accurately classify every patient under the 2 classes ‘Will be Readmitted’ and ‘Will not be Readmitted’ by building predictive machine learning/deep learning models which are capable of doing binary classification. Through our methods, processes and modelling techniques we were able to accomplish this goal with a 64% accuracy in our prediction, i.e. out of 100 patients, 64 of our patient predictions are most likely going to be correct.

As a future scope we can use a more advanced spark architecture with higher memory and computational resources in order to train our computationally intensive ML algorithms on the entire train dataset instead of training it on a smaller subset. We can also explore more advanced feature selection techniques and more advanced learning algorithms that can perform better than our current models and produce a higher Accuracy & Recall.