

**Predicting Readmissions within 30 days**

HARIKA POLAKI

**Table of Contents**

1. **Introduction**
2. **Problem Statement/ Project Definition**
3. **About Dataset**
4. **Data Pre-processing**
   1. **ADMISSIONS**
   2. **DIAGNOSIS\_ICD**
   3. **PATIENTS**
   4. **NOTEEVENTS**
5. **Natural Language Processing**
6. **Splitting the dataset**
7. **Model Selection**
   1. **Decision Trees**
   2. **Logistic Regression**
   3. **Stochastic Gradient Descent**
   4. **Naïve Bayes**
   5. **Gradient Boosting**
   6. **KNN**
   7. **Random Forest**
8. **Feature Importance**
9. **Hyper Tuning Parameters**
10. **Best Model**
11. **Conclusion**
12. **My Learning**
13. **References**

**1 Introduction**

In this modern era of increasing technology, many changes have taken place in the healthcare industry. The Centers for Medical and Medicaid Services(CMS) brought some revolutionary changes to provide better healthcare. CMS has created various programs to improve the quality of care for the patients. Once such a revolutionary program is called the Hospital Readmission Reduction Program(HRRP), this program's main aim is to reduce reimbursement to hospitals with above-average re-admissions. The primary objective of the hospitals currently enrolled under this program is to create interventions to provide additional assistance to patients with increased risk of re-admission. Hospitals use predictive analysis for filtering out these requirements. Predictive analytics is an essential tool in the healthcare industry since modern machine learning methods can use large amounts of available data to predict patients' outcomes.

The project's main aim is to develop a model for predicting whether a patient discharged from a hospital is likely to readmit within 30 days. The project makes use of the MIMIC database. MIMIC is a critical care database developed by MIT lab for Computational Physiology, comprising identified health data associated with ~60,000 essential care patients.

**2 Project Definition / Problem Statement**

Predict the probability of patient's re-admission within 30 days of discharge based on the patient's release records upon discharge.

Solution outline to build the model:

* Data Exploration
* Data Preprocessing
* Building Training/ Test samples
* Model Selection
* Model Evaluation based on the metrics

**3 About dataset**

MIMIC III is a relational database consisting of 26 tables. Tables are linked by identifiers, which usually have the suffix 'ID.' For this project, four datasets, namely ADMISSIONS, DIAGNOSIS\_ICD, NOTEEVENTS, PATIENTS, have been used. The database includes information about procedures, medication, lab test results, mortality, imaging reports, and much more.

* ADMISSIONS: This dataset contains information regarding a patient's admission to the hospital covering an admission period between June 1, 2001, to October 10, 2012, and has a unique feature called HADM\_ID unique ID given to the patient during their visit to the hospital. The information available includes the patient's timing information, discharge, demographic information, the admission source, and so on. The dataset consists of 58796 rows.
* DIAGNOSIS\_ICD: This database contains ICD diagnoses for patients, mainly ICD-9 diagnosis codes. These codes are generated for billing purposes at the end of the hospital stay. ICD\_9 CODE includes the actual code corresponding to the diagnosis assigned to the patient. The entire dataset consists of 651047 rows.
* PATIENTS: This dataset contains patient information like unique SUBJECT\_ID, gender, DOB. SUBJECT\_ID is a unique identifier that specifies an individual patient. It has 46520 rows.
* NOTEEVENTS: This dataset contains all notes for patients. It has information about discharge events, which condense information about a patient's stay into a single document. It contains 2083180 rows.

**4 Data Preprocessing**

Data preprocessing is generally used to select the appropriate/ needed data to apply machine learning algorithms. It is applied to make data more useful for data mining. Data processing is about selecting relevant data objects and attributes to analyze and create or change the attributes. Here the preprocessing has been done separately for respective datasets.

* 1. **ADMISSIONS Dataset**

Converted all dates into Date Time format. Setting the errors=’coerce' flag allows for missing dates, but it puts it to NaT when a particular string doesn't match the format.

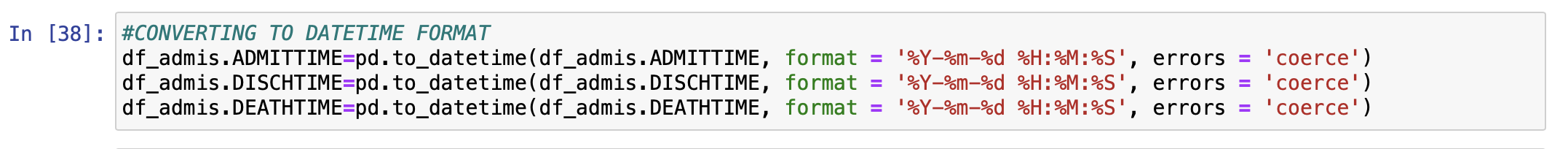


Fig 1

Next, verify that the dates are in order. Later use the Shift() function to get the next admission date. Use group and shift operator to get the following admission for each SUBJECT\_ID. Since the prediction here is about unplanned re-admissions, the next step is to drop entries which are ELECTIVE so that only emergency re-admissions are measured.

Text

Description automatically generated

Fig 2

Next, calculating the days between discharge and the next emergency visit. Now count the total number of re-admissions in the dataset. Currently, there are 58976 admissions in total and 11399 total re-admissions. For those with re-admissions, a histogram has been plotted for days between admissions.

Graphical user interface, text, application, Word

Description automatically generated

Fig 3

Chart, histogram

Description automatically generated

Fig 4

Later compressed the Ethnicity categories.

Text

Description automatically generated

Fig 5

* 1. **DIAGNOSES\_ICD Dataset**

There is a total of 6984 unique ICD9 codes in this particular dataset.



Fig 6

Later reduced the diagnosis codes into general categories and converted diagnosis list codes into hospital admission-item matrix.

Background pattern, table

Description automatically generated

Fig 7

Concatenated to ADMISSIONS CSV on HADM\_ID

Table

Description automatically generated

Fig 8

* 1. **PATIENTS Dataset**

Converted date of birth into Date Time format. Gender feature is binarized. Calculated age using the difference between DOB and date of admission.

Table

Description automatically generated

Fig 9

Chart, histogram

Description automatically generated

Fig 10

* 1. **NOTEEVENTS Dataset**

There is a total of 2,083,190 notes. The number of notes is much higher than the number of hospitalizations, which means there are multiple notes per hospital. SUBJECT\_ID, HADM\_ID, CHARTDATE, CATEGORY, TEXT are the primary columns of interest. 11% of the notes are missing HADM\_ID's. HADM\_ID is missing for Outpatient- An outpatient is a patient who receives medical treatment without being admitted to the hospital—used only discharge summary by filtering out 'Discharge Summary.'

Text

Description automatically generated

Fig 11

**Final Dataset**

After all the preprocessing, produced a clean CSV with 45059 records and 3018 attributes. A histogram has been built where class 1 represents patients readmitted within less than 30 days, and class 0 represents other than that. There are 2957 records and 42012 records for each category, respectively.

Chart, bar chart

Description automatically generated

Fig 12

**5 Natural Language Processing**

Natural Language Processing (NLP) is a subfield of Artificial Intelligence. It helps the machine process and understands the human language in any context. NLP's sole objective is to read, decipher, understand, and make sense of human languages. The following steps were applied in the project:

* Transforming all text to lower case
* Removing numerical
* Removing Punctuations
* Removed stop words used
* Tokenizing- It is a step where longer strings are breaking down into smaller tokens. Tokenized larger chunks of data into sentences and sentences into words.
* Stemming- It is a process of eliminating affixes from a word to obtain a word stem.
* Lemmatization- Similar to stemming, the difference is that it can capture canonical forms based on the word's lemma. Using lemmatization, inflectional endings can be removed and return to the base form of the word.
* Count Vectorization- It is a very basic preprocessing like removing punctuation marks, converting all the words to lowercase, etc.

Graphical user interface, text, application, email

Description automatically generated

Fig 13

Graphical user interface, text, application

Description automatically generated

Fig 14

**6 Splitting the dataset into Training and Testing datasets**

Any ML algorithm needs to be tested for accuracy. To do that, the dataset needs to be divided into training and test set. The training set is to train the algorithm, whereas the test set is used to check the algorithm's correctness. Here 70% of data is under training set, and in the remaining 30%, 15% as the validation set, and the other 15% as the test set. And now split validation and test set using 50% fraction.

As this is an imbalanced dataset, and there are very few positive cases, the dataset is divided into 50% positive and 50% negative cases.

Graphical user interface, text, application

Description automatically generated

Fig 15

**Random Forest:**

Using a Random Forest classifier with 300 estimators, generated a model. The results are as follows:

Text

Description automatically generated

Fig 16

Chart, treemap chart

Description automatically generated

Fig 17

Chart, line chart

Description automatically generated

Fig 18

The current dataset is imbalanced as the number of negative samples is more than positives, so the model might assign all samples negative.

Text

Description automatically generated

Fig 19

To balance the imbalanced data, there are few techniques like

* Subsample the more dominant class
* Oversample the imbalanced data
* Create Synthetic positive data (E.g., SMOTE)

Oversampling and Undersampling are known as Resampling. When one class of data is under-represented in the data sample, oversampling techniques can duplicate these results to obtain a more balanced data sample. SMOTE is a technique used to create synthetic data by randomly sampling the characteristics. When a category is over-represented in the data sample, under-sampling is used to balance the dataset. In both oversampling and sampling, simple data duplication is rarely used. Under-sampling can result in loss of data sometimes. Here Oversampling has been implemented, and later Random Forest Classifier has run. The performance metrics are as follows:

Text

Description automatically generated

Fig 20

Chart, waterfall chart

Description automatically generated

Fig 21

Chart, line chart

Description automatically generated

Fig 22

**Effect of Estimators**

Text

Description automatically generated

Fig 23

Chart, line chart

Description automatically generated

Fig 24

Chart, scatter chart

Description automatically generated

Fig 25

Setting estimators' value as 300 has given better results. It is better to continue with such a low amount as we can save time and cost.

**Feature Selection**

Feature selection is the process where you select features that contribute most to your output. Having irrelevant features in data can reduce accuracy. The advantages of Feature Selection are

* Reduces Overfitting
* Improves Accuracy
* Reduces Training Time

The feature selection method used in this project is the Chi-Square test. A statistical test is used to determine whether the output variable is dependent or independent of the input variable. The calculations are done between the variables and the target. If there is a dependency between both, then that particular variable is included; otherwise, it is excluded.

Graphical user interface, text, application, email

Description automatically generated

Fig 26

Chart, scatter chart

Description automatically generated

Fig 27

The above Random Forest Classifier has been implemented after Feature Selection. The n value for estimators has been reduced to 200 from 300. The model has performed well than before.

**7 Models Selection/ Implementation**

In this section, few machine learning models have been selected and implemented, and the best model is determined based on the performance of the validation set.

* 1. **Decision Tree**

Tree-based methods are popular ML models. The technique behind decision tree implementation is the samples are split behind a certain threshold. The advantage of tree-based methods is that they have no assumptions about the data structure and can pick up non-linear effects if given sufficient depth. The decision tree is fitted using the following code. Different metrics, like AUC, accuracy, recall, precision, specificity, and prevalence, have been calculated. This has been done for all the models that were implemented.

**Graphical user interface, text, application, email

Description automatically generated**

Fig 28

* 1. **Logistic Regression**

Logistic Regression is a traditional ML model that fits a linear decision boundary between positive and negative samples. The model is easily interpretable. Logistic Regression is fit using the following code:

**Graphical user interface, text, application, email

Description automatically generated**

Fig 29

* 1. **Stochastic Gradient Descent**

Stochastic gradient descent is similar to logistic Regression. Both methods use gradient descent to optimize the coefficients of a linear function. In SGD small batch of samples is used. Because of this, SGD will speed up its process.

**Graphical user interface, text, application, email

Description automatically generated**

Fig 30

* 1. **Naïve Bayes**

In Naïve Bayes, Bayes Rule is used to calculating the probabilities. The model assumes that all the features are independent.

**Graphical user interface, text, application, email

Description automatically generated**

Fig 31

* 1. **Gradient Boosting**

Many shallow trees have been created in boosting that try to improve the previously trained trees' errors. The model that uses this technique paired with the Gradient descent algorithm is called as Gradient Boosting Classifier.

**Graphical user interface, text, application, email

Description automatically generated**

Fig 32

* 1. **K nearest neighbors**

It is one of the simplest ML models where the model takes the closest k data points and determines the probability by counting the number of positive cases divided by k.

Graphical user interface, text, application, email

Description automatically generated

Fig 33

* 1. **Random Forest**

Random Forest is created to reduce overfitting. Here multiple trees are made, and the results are aggregated. In most cases, random forests work better than the decision tree as they can generalize more easily.

Text, letter

Description automatically generated

Fig 34

A data frame with the results of all the models is prepared to compare the models. AUC is used to evaluate the best model because it captures the trade-off between true positive and false-positive and doesn't require selecting a threshold.

**Chart, bar chart

Description automatically generated**

Fig 35

From the above graph, it is clear that most of the models have similar performance on the validation set. The significant drop between training and validation sets is because of overfitting.

**8 Feature Importance**

Top 50 positive features have been predicting from Random Forest.

Chart

Description automatically generated

Fig 36

This plot helps concentrate on essential features. In the case of high variance, overfitting can be reduced by reducing the number of variables. For analysis, using top N features would be sufficient. Through feature importance plots, errors in the predictive model can be pointed out.

**9 Hyper Tuning for Model Selection**

Through Hyper tuning parameters can be optimized to improve the model. Hyperparameter tuning is the design decision that you made when the ML model is setup. Here optimization for hyperparameters in for SGD, Random Forest, Gradient Boosting, and Naïve Bayes. One technique for hyperparameter tuning is Gris Search, where testing of all possible combinations over a grid of values is done. Another option is RandomSearch.

Hyper Tuning – Random Forest

Text

Description automatically generated

Fig 37

Hyper Tuning-SGD

Text

Description automatically generated

Fig 38

Hyper Tuning-Gradient Boosting

Graphical user interface, text, application, email

Description automatically generated

Fig 39

Hyper Tuning-Naïve Bayes

Graphical user interface, text, application, email

Description automatically generated

Fig 40

Aggregate the results and compare them to the baseline models on the validation set.

Chart, bar chart

Description automatically generated

Fig 41

From the above graph, it is clear that hyper tuning improved the models. There is no much difference because of the High Bias.

**10 Best Model**

Gradient Boosting is the best model for predicting re-admissions because it has given the best AUC among all.

Evaluating Metrics of the Best Model:

Graphical user interface, text, application, email

Description automatically generated

Fig 42

Chart, line chart

Description automatically generated

Fig 43

**11 Conclusion**

Many ML models have been implemented throughout this project to predict a Patient's Re-admission within 30 days. Gradient Boosting stood as the best model classifier with optimized hyperparameters. The model was able to catch 58-59% of the re-admissions. Since data was imbalanced hyper tuning increased the accuracy of the models.

**12 My Learning From This Project**

As a student from the Engineering department, I got a chance to explore Health Care Dataset through this course and project. Implementing ML algorithms on health dataset has been an amazing experience. Learned a lot of medical terms through this course.

**13 References**

1. [**https://towardsdatascience.com/predicting-hospital-readmission-for-patients-with-diabetes-using-scikit-learn-a2e359b15f0**](https://towardsdatascience.com/predicting-hospital-readmission-for-patients-with-diabetes-using-scikit-learn-a2e359b15f0)
2. [**https://medium.com/nwamaka-imasogie/predicting-hospital-readmission-using-nlp-5f0fe6f1a705**](https://medium.com/nwamaka-imasogie/predicting-hospital-readmission-using-nlp-5f0fe6f1a705)
3. [**https://www.kaggle.com/questions-and-answers/141025**](https://www.kaggle.com/questions-and-answers/141025)
4. [**https://www.featureranking.com/tutorials/machine-learning-tutorials/sk-part-3-cross-validation-and-hyperparameter-tuning/#1.5**](https://www.featureranking.com/tutorials/machine-learning-tutorials/sk-part-3-cross-validation-and-hyperparameter-tuning/#1.5)