Predicting Diabetes Patients Early Readmission using ML models

Roni Weinfeld [1], Maya Sheffer [1], Thea Meimoun [1] and Shachaf Frenkel[1]
[1] Weizmann Institute of Science 234 Herzl Street, POB 26, Rehovot 7610001 Israel
(Dated: May 12, 2024)

This machine learning project aimed at forecasting patient readmission rates utilizing the 'Diabetes 130-US Hospitals for Years 1999-2008' database [1]. The performance of four ensemble algorithms: LightGBM, XGBoost, RandomForest, and CatBoost, is compared within this work.

I. INTRODUCTION

A. Diabetes Medical Overview

Diabetes, a chronic condition, is characterized by heightened blood glucose levels, along with disrupted metabolism of fats and proteins. This elevation in blood glucose occurs because it cannot be properly metabolized within cells, either due to insufficient insulin production by the pancreas or the cell's ineffective utilization of the produced insulin. The condition encompasses three primary types: (1) Type 1, where the pancreas fails to produce insulin; (2) Type 2, where the body cells become resistant to the insulin produced, leading to a gradual decrease in insulin production over time; and (3) gestational diabetes, occurring during pregnancy, which can result in complications during pregnancy and childbirth, and increases the risk of type 2 diabetes in the mother as well as obesity in the offspring. [2].

B. Medical Relevance of the Features

All features are medically relevant for the data's intended purpose, which is to predict whether a diabetic patient will be readmitted to the hospital within the next 30 days after admission.

a. Patient Demographics: Information such as age, gender, race, and socioeconomic status provides insights into how these factors influence diabetes prevalence, progression, and treatment outcomes. Understanding demographic trends can aid in tailoring interventions and healthcare policies to address disparities in diabetes management and outcomes among different population groups.

b. Clinical Characteristics: This includes parameters like body mass index (BMI), blood pressure, glycemic control measures (e.g., HbA1c levels), and presence of comorbidities (e.g., hypertension, cardiovascular disease). These factors are crucial for assessing disease severity, risk stratification, and guiding treatment decisions in individuals with diabetes.

c. Treatments: Information on medication usage, insulin therapy, and surgical interventions offers insights into the effectiveness of different treatment modalities in controlling blood glucose levels, preventing complications, and improving overall patient well-being. Comparative analysis of treatment strategies can inform evidence-based guidelines and optimize diabetes management protocols.

d. Hospital Operational Factors: Variables such as hospital size, location, and resources available play a role in determining access to care, quality of treatment, and patient outcomes. Understanding the impact of hospital-related factors on diabetes management can inform healthcare resource allocation, facility planning, and quality improvement initiatives aimed at enhancing the delivery of diabetes care services.

e. Outcomes: Outcome measures such as readmission rates, length of hospital stay, mortality, and incidence of complications (e.g., diabetic ketoacidosis, hypoglycemia) provide insights into the effectiveness of healthcare interventions, disease management practices, and healthcare system performance in addressing the challenges posed by diabetes. Monitoring these outcomes facilitates continuous quality improvement efforts and helps in evaluating the impact of interventions aimed at reducing the burden of diabetes-related morbidity and mortality.

Overall, the comprehensive assessment of these features within the dataset enables a multifaceted understanding of diabetes epidemiology, management, and outcomes, thereby informing evidence-based strategies for improving the quality of diabetes care and reducing the burden of this chronic disease on individuals and healthcare systems alike. However, when considering data quality and the unique information provided by each feature, there are a few features that could be excluded from the dataset or combined into new, more relevant features.

C. Literature Review

Hospital readmission prediction has been extensively explored through the lens of machine learning algorithms in various scholarly works. The prominent algorithms employed include Random Forest, XGBoost, Support Vector Machine (SVM), and Decision Tree - Elazhary and Alajmani [3] leveraged Random Forest and SVM, while Dixit [4] utilized Decision Tree for this purpose. An overarching consensus among these studies is the identification of key predictors associated with

hospital readmission. Notably, the frequency of emergency room visits in the year preceding hospitalization emerged as the primary predictor across all algorithms. Following closely, the number of lab procedures conducted during the hospital stay was identified as the second most influential factor. However, the third most significant predictor varied among the algorithms employed [5].

Pre-processing datasets is vital for ensuring the robustness of predictive models, primarily due to challenges such as data noise and imbalanced labels [6]. To mitigate these issues, researchers often resort to techniques such as log transformation to normalize skewed data and handle missing values by imputing them with mean or mode values. Additionally, various strategies for feature selection are employed, ranging from selecting features based on their importance to engineering new features [7].

Addressing the imbalance in data distribution is imperative to prevent bias and overfitting. Techniques such as bagging, involving resampling with replacement, are commonly recommended. Moreover, both over and under-sampling methods are employed, with some studies utilizing synthetic data generation techniques like Generative Adversarial Networks (GAN) and Synthetic Minority Over-sampling Technique (SMOTE) [8] [9].

In summary, the literature underscores the significance of machine learning algorithms in predicting hospital re-admissions. By identifying relevant predictors and employing appropriate preprocessing and handling techniques for imbalanced data, these algorithms offer valuable insights for healthcare practitioners in managing patient care effectively.

D. Metadata Description of the Dataset

The dataset is used for studying and analyzing diabetes-related trends and outcomes in US hospitals from 1999 to 2008. It contains 101766 instances and 47 features:

- Encounter ID: unique identifier of an encounter.
- Patient number: unique identifier of a patient.
- Race: Caucasian, Asian, African American, Hispanic, and other.
- Gender: male, female, and unknown/invalid.
- Age: grouped in 10-year intervals: 0, 10), 10, 20), . . . , 90, 100)
- Weight: weight in pounds.
- Admission type: integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available.
- Discharge disposition ID: integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available.
- Admission source: integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital.
- Time in hospital: integer number of days between admission and discharge.
- Payer code: integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay Medical.
- Medical specialty: integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon.
- Number of lab procedures: number of lab tests performed during the encounter.
- Number of procedures: numeric Number of procedures (other than lab tests) performed during the encounter.
- Number of medications: number of distinct generic names administered during the encounter.
- Number of outpatient: visits Number of outpatient visits of the patient in the year preceding the encounter.
- Number of emergency: visits Number of emergency visits of the patient in the year preceding the encounter.
- Number of inpatient: visits Number of inpatient visits of the patient in the year preceding the encounter.
- Diagnosis 1: the primary diagnosis (coded as first three digits of ICD9); 848 distinct values.
- Diagnosis 2: secondary diagnosis (coded as first three digits of ICD9); 923 distinct values.
- Diagnosis 3: additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values.
- Number of diagnoses: number of diagnoses entered to the system.
- 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.

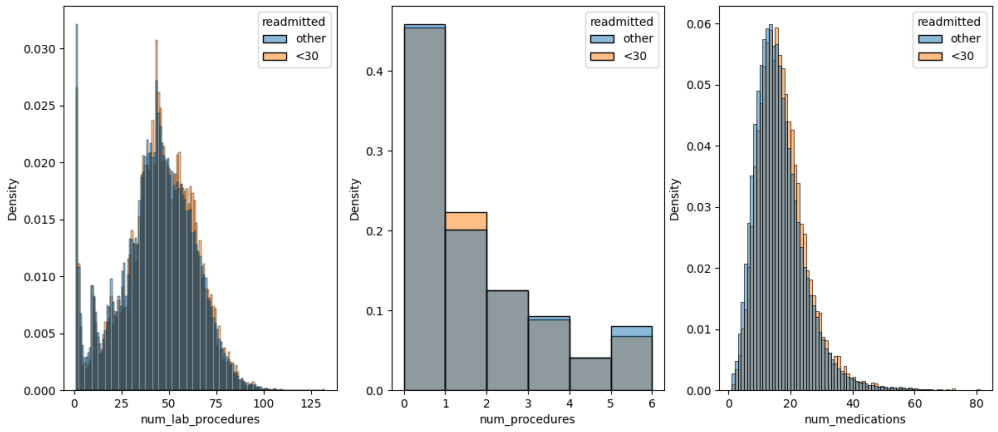

FIG. 1: Histograms of several features- with respect to the readmission rate of each value. We can see a slight difference between the early readmitted group and the late or no readmission group.

- 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.
- Change of medications: indicates if there was a change in diabetic medications (either dosage or generic name). Values: "change" and "no change".
- Diabetes medications: indicates if there was any diabetic medication prescribed. Values: "yes" and "no" 24 different kinds of medical drugs.
- Readmitted: days to inpatient readmission. Values: "0" 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.

E. Objective

Our primary objective is to forecast the likelihood of diabetic patients being readmitted to the hospital subsequent to their initial visit and treatment. The focal point of our prediction is the label 'readmission', which in the original dataset consists of three distinct categories: '¡30' (indicating readmission within 30 days), '¿30' (representing readmission after more than 30 days), and 'NO' (indicating no readmission occurred). The features encompass all columns present in the dataset prior to any pre-processing steps. Subsequently, we will detail the alterations made to the data based on conclusions drawn from exploratory data analysis (EDA) and feature engineering.

II. DATA PREPARATION AND EXPLORATORY DATA ANALYSIS

A. Train and Test Sets Split

The data was split into train and test sets using the $train_test_split$ function from the sklearn library, with patient ID numbers used as the criteria to ensure that each patient is represented only in either the train or test set, even if there are multiple records for a patient. Additionally, a statistical check was performed to ensure that gender and readmission rates are balanced between the train and test sets.

The classification label within the 'readmitted' feature underwent redefinition: instances of 'NO' readmission and readmission occurring after 30 days ('> 30') were assigned the value of 0, denoting absence of readmission, while instances labeled as '> 30' were categorized as readmitted within 30 days. This categorization approach was selected based on its economic and clinical significance: reducing the rate of early re-admissions in hospitals holds potential benefits for both institutions, in terms of cost savings associated with subsequent hospitalizations, and patients, by averting potential mistreatment. However, this method of label grouping exhibits robust imbalance, with only 10.4% of instances denoting readmission across the entire training dataset.

B. Data Quality

a. Assessment of missing values: There are missing data points in seven features. Therefore, a missing value percentage was calculated for each feature to establish a threshold for omitting missing values in a manner that minimizes data loss.

b. Unique values variance : scrutiny of unique values across features unveiled imbalanced distributions, indicating certain features lacked distinctive information, potentially rendering them irrelevant for predictive modeling. Additionally, key demographic features such as age, gender, and admission status exhibited imbalance, necessitating tailored strategies during model training to mitigate biases.

c. Feature correlation with the label: a heatmap visualization (fig. 2a) was employed to examine correlations between numerical features and the target label. Notably, the number of outpatient visits emerged with the highest correlation to the label and a notable correlation was observed between the number of medications and the duration of hospitalization, hinting at potential associations between healthcare utilization patterns and clinical outcomes. The association between categorical features and the label was illustrated through the visualization of the normalized readmission rate per unique value within each categorical feature (Figure 2b). This analysis revealed compelling trends pertaining to certain prescribed medications and their correlation with readmission rates, as well as the influence of attending physicians on patient readmission.

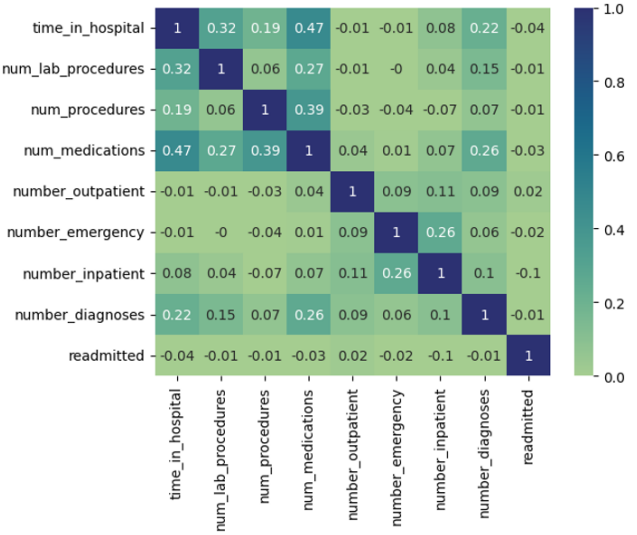

FIG. 2: Assessment of Relationships between Features and Readmission Rates: Normalized Pearson's correlation between numerical features and the label.

d. Outliers: Data outliers evaluation was conducted using comparison between the percentile omission and the data loss percentage. Upon this a 10 percentile was chosen for omitting numeric data outliers with the minimal data loss.

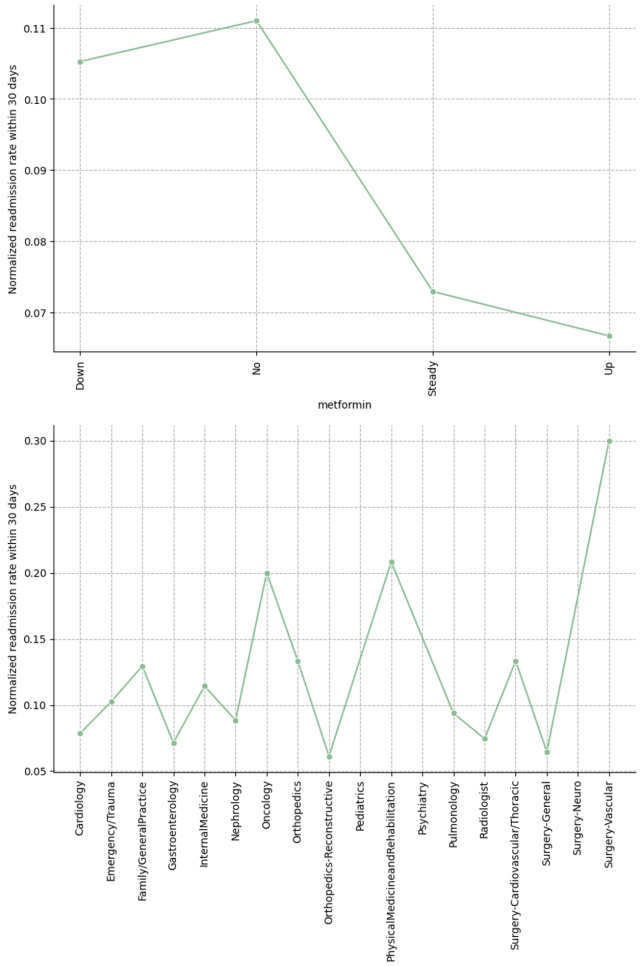

FIG. 3: Assessment of Relationships between Features and Readmission Rates: Normalized readmission rate per unique value in a feature. The readmission rate was calculated by dividing the count of re-admissions within 30 days by the count of the unique value.

C. Data Challenges

The potential challenges in the dataset mostly involve missing values and imbalanced distribution of the label. The missing values account for 3.8% of the dataset - For instance, the 'weight' feature contains 97% missing information. Additionally, some features exhibit low variance among their unique values (fig 1). For example, the features 'number outpatient', 'number emergency', 'miglitol', 'troglitazone', and 'metformin' demonstrate more than 95% of the same value. There are no significant ethical issues with the database; however, atten-

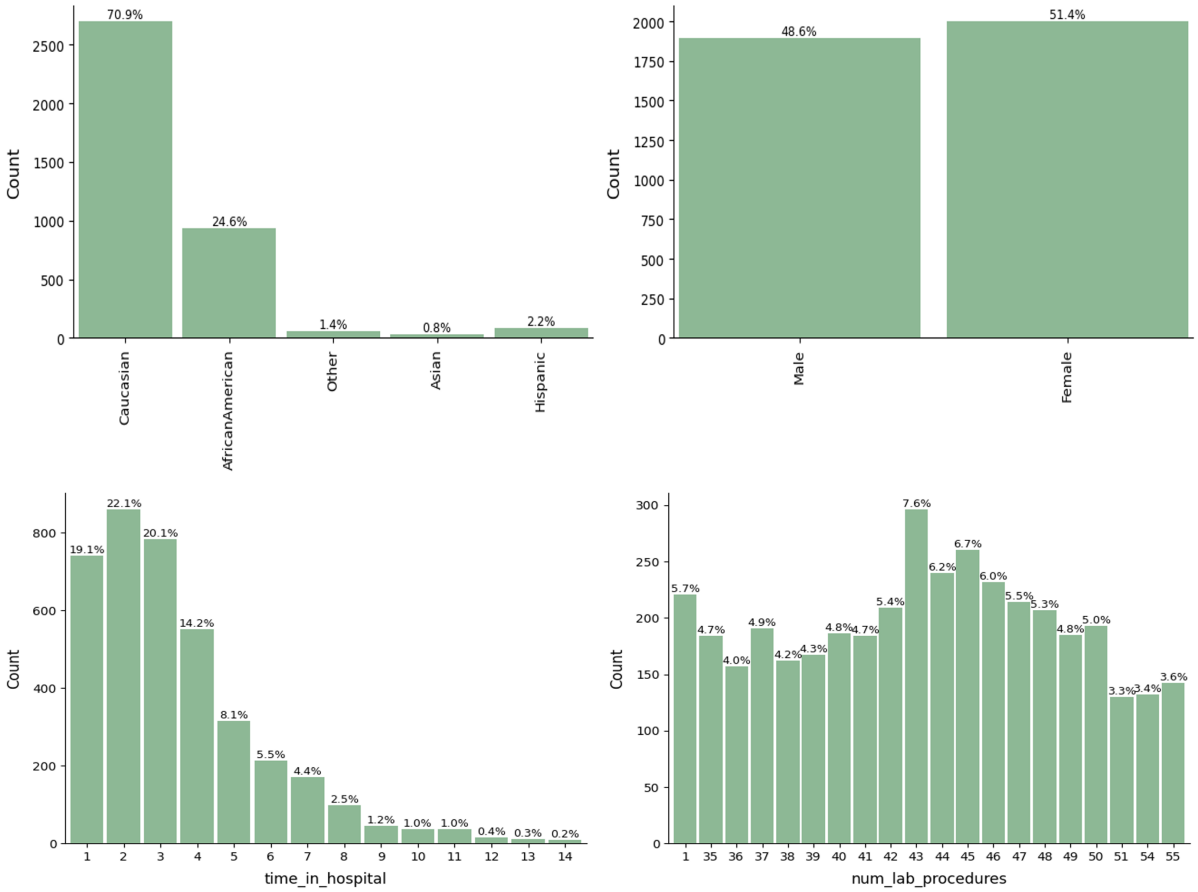

FIG. 4: Histograms of several features- numerical and categorical.

tion must be given to the racial distribution. Approximately 70% of the patients are Caucasian, suggesting that research findings may not be applicable to groups underrepresented in this dataset.

D. Data Preparation

The initial data assessment confirmed that the dataset adhered to the principles of tidy data, with each variable allocated to its own column, every observation represented as a separate row, and distinct types of observational units organized into individual tables. Additionally, no data duplications were detected.

However, to optimize the dataset for subsequent analysis, several preprocessing steps were undertaken. Initially, non-informative features such as 'payer code' were eliminated. Additionally, columns with a significant proportion of missing values exceeding 10%, such as 'weight', were removed. Furthermore, features lacking variance in their values, such as 'acetohexamide', 'tolbutamide', 'troligtazone', 'tolzamide', 'examide', 'citoglipton', 'glipizide-metfomine', 'glimepiride-pioglitazone', 'metformin-rogiglitazone', and 'metformine-pioglitazone', were also discarded.

Subsequently, feature engineering procedures were implemented to enhance the dataset's interpretability and predictive power. Values with similar attributes within the 'medical specialty' feature were consolidated to reduce redundancy. Moreover, categorical variables such as 'discharge_disposition_id' were reorganized into broader categories, yielding four distinct features: 'death', 'home', 'hospital', and 'left_ama'. Additionally, a new feature, 'admission_type_id', was introduced to classify admission severity into 'elective' or 'emergency' based on the values of the 'admission_source_id' feature.

Finally, to ensure optimal model performance and interpretability, a series of preprocessing techniques were applied. Numerical features underwent feature scaling to standardize their scales, facilitating fair comparison and model convergence. Ordinal features were subjected to ranking to preserve their inherent order while nominal features were transformed using one-hot encoding to convert categorical variables into a format suitable for model training. Additionally, missing values within the dataset were imputed using appropriate

techniques tailored to the specific feature and model requirements. These comprehensive preprocessing steps collectively contributed to enhancing the robustness and effectiveness of the subsequent modeling process.

E. techniques to account for the imbalance in the dataset

Since the readmission rates in the dataset vary and the minority class (10% of the dataset) is the label we are interested in, we had to balance the dataset. We chose to implement various undersampling techniques (random under sampling and edited nearest neighbors) to undersample the majority class and oversampling techniques (SMOTENC, CopulaGAN, CTGAN) to oversample the minority class. For all algorithms except those that use a GAN we used the Imbalanced Learn library during the cross-validation process, only on the training data. This was done to keep the validation set as "neutral" as possible, so it will truly represent the test set or any other new data that will be used as input to test our model.

a. Undersampling A significant number of the features in our dataset are categorical and some learning algorithms we used perform better WITHOUT preprocessing with one-hot encoding (like LightGBM with integer-encoded categorical features- LightGBM applies Fisher [10] to find the optimal split over categories). Giving as an input categorical data and not as OHE form and manipulating it is not trivial, therefore we could use on the non-OHE data only random under sampling of the majority class. Random under sampling simply samples randomly rows that have a label that matches the majority class label.

In order to try more sophisticated undersampling methods, we had to apply OHE to the data and therefore not use the built-in ability of our learning algorithms to deal with categorical non-ordinal data. We used an edited nearest neighbors algorithm which cleans the dataset by removing samples close to the decision boundary. It removes observations from the majority class when any or most of its closest neighbors are from a different class - the minority, in our binary classification problem. We found that randomly under sampling yields better performance measures for the validation set and therefore we chose this form of undersampling. This observation might be a result of noisy data in the validation set - cleaning the training set only made it differ from the validation set and therefore the model that fits well the training set doesn't necessarily fits well the validation set.

b. Oversampling In contrast to undersampling, oversampling can be more easily done on non-OHE categorical data with designated algorithms. One such algorithm is SMOTE-NC (Synthetic Minority Oversampling Technique for Nominal and Continuous), a variation of the SMOTE algorithm. This technique was described by Nitesh Chawla, et al. in their paper named for the technique titled "SMOTE: Synthetic Minority Over-sampling Technique" [11]. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

In addition, we synthesized new samples with GANs - including CTGAN and CopulaGAN with the SDV library. We trained the GAN only on the minority class (which is only 10K samples). Generally, a GAN is actually two neural networks that contest with each other - i) the generator- needs to generate new synthetic samples. ii) the discriminator- needs to distinguish between true samples and synthetic ones (created by the generator). After some iterations (epochs) the GAN learns to generate new data with similar statistics as the training set.

We trained both GANs with default parameters but with more epochs (500), thanks to the use of GPU to speed-up the training process. We tried several batch-sizes but the result of the evaluation test of the SDV library didn't vary, so we chose the default batch size. As discussed in the SDV developers blog [12], the generator's loss should decrease but the discriminator's loss should oscillate around zero. Indeed, as seen in figure 5, both losses behave as they should for a proper learning process.

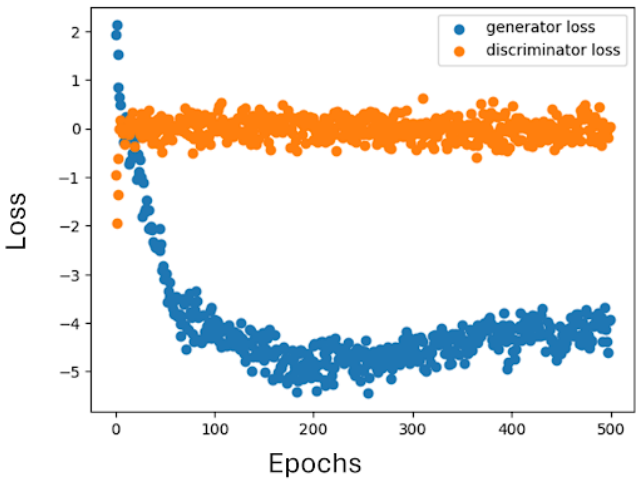

FIG. 5: The generator's and discriminator's loss as a function of epochs. The oscillation around zero of the discriminator's loss and the decrease of the generator's loss is similar to the typical behavior described by the CopulaGAN developers [12].

F. Performance Measure

We selected 'balanced accuracy' as our primary performance metric and used a 'confusion matrix'. This choice stems from the highly imbalanced nature of the dataset, where there exists a significant 80% to 20% disparity between the labels. Consequently, traditional accuracy, learning rate, and logloss scores may appear favorable, despite the model's tendency to merely classify instances into the majority class label without effectively learning from the data.

G. Code Overview

The code consists of three main files: classes.py, preprocessing.py, and hyperparameter_tuning_and_evaluation.py, each serving distinct purposes in the framework of the project. In classes.py, custom transformer classes are defined using sklearn base classes to match sklearn pipelines and offer flexibility in data preprocessing, tailored to the specific requirements of the dataset. These transformers include dropping features based on variance, grouping of similar categories, and more. preprocessing.py contains different pipelines that perform the preprocessing in steps. The script can execute preprocessing for different uses - (1) basic transformation and data handling for generating synthetic data using GAN and SMOTE algorithms, (2) advanced preprocessing after adding the synthetic data and combining it into one data frame ready for cross-validation. Option to use different steps depending on the model's requirements, (3) preparing the data for final evaluation over the test set. The next script split the dataset into training sets and validation sets while ensuring that patients are either in the training set or the testing set but not in both. After splitting the dataset, the code synthesizes new examples using the GAN or SMOTE base in the training folds. The output of the code is a new dataset that contains 10 folds with synthetic training examples. This part of the code was run with COLAB's GPU. Then the last script, model_training_and_evaluation.py, implements the hyper-parameter tuning using grid search and OPTUNA and evaluates the results over different seeds, on BalancedRandomForest and catBoost models. In addition, we did hyper-parameter tuning for lightGBM and XGBoost in separate files.

H. fine Tunning of the pre-processing

Fine-tune the preprocessing using the BalancedRandomForest classifier First, we divided the preprocessing steps into 7 independent steps: Convert age from nominal scale to ordinal scale. Group relative diagnosis (diag_1,diag_2,diag_3) using domain knowledge into10 categories - 'circulatory', 'respiratory', 'digestive', 'diabities_without_complications', 'diabities_with_complications', 'injury', 'musculoskeletal', 'genitourinary', 'neoplasms', 'pregnancy' and 'other'. Group together relative medical specialty ('medical_specialty' feature) using domain knowledge into 5 main groups 'surgery', 'neurology', 'psychic', and 'pediatrics'. Also, we grouped all the categories that had low frequency in the dataset into a new category called 'low_frequancy' Group together relative symbols from 'admission_source_id', 'admission_type_id', and 'discharge_disposition_id' by the meaning of each symbol as reported by Beata Strack et al., (2014). Drop low variance features. Drop outliers, patients. We cross-validate (n folds = 10) the classifier with default parameters, except for max depth which was set at 25 in order to regulate the model and minimize the overfitting. We also used. under-sampling of 4:1 ratio between the classes. over each step, we measured the balanced accuracy over the validation and training sets. We chose the mean balanced accuracy over the validation folds to be our evaluation method in this context. Using this method we found that steps1,2,3,5 gave the best results, so we continued only with those steps and reached a mean balanced accuracy of 0.5856 over the test set. Later we decided to continue to do feature engineering and evaluate the quality of the manipulations using the same method described above. We found that the following steps increase the balanced accuracy over the validation folds:

- Severity of the disease feature: $Severity\_of\_disease = 2 * time\_in\_hospital + num\_procedures + num\_medications + num\_lab\_procedures$
- Sickness index feature: $Sick_index = number\_emergency + number\_inpatient + number\_outpatient$
- Manipulate the 'change' feature by replacing 'Ch' with 1 and -1 for other values. This manipulation converts this feature from categorical to numerical.
- Drop all patients that belong to the 0-10 age group.

Furthermore, we look at the feature importance score of the different models and tried to eliminate the less important features. This step did not have effect on the cross- validation scores. After we finish tuning the models we evaluate the combination of the features over the prediction of the labels 8.

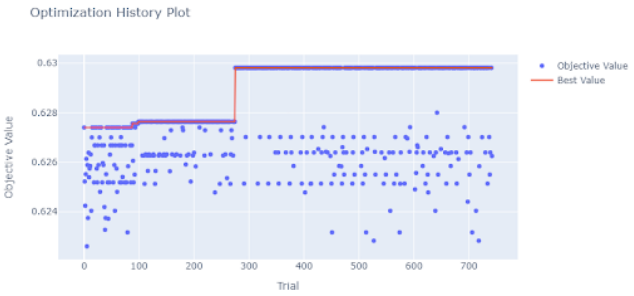

FIG. 6: The balanced accuracy as a function of trial number, during the Optuna hyper-parameters tuning process.

III. CLASSIFICATION MODELS

A. XGBoost - Thea Meimoun

XGBoost, eXtreme Gradient Boosting, based on the principles of gradient boosting, improves model predictions, mainly decision trees, through optimization. Each successive tree is built to correct previous errors in the existing set, then redefine the overall predictive accuracy across all iterations. XGBoost proposes regularization methods, as L1 and L2, to help reduce model complexity and the risk of overfitting. This standardization is essential to maintain the model's robustness against data variance. XGBoost offers advanced features that differentiate it from traditional boosting methods:

- Tree pruning: XGBoost uses a "depth-first" approach to developing and building trees, before the whole tree is built. This method allows the algorithm to stop adding branches when it determines additional splitting will not lead to significant learning gains. This proactive pruning optimizes computing resources by avoiding unnecessary computation and complexity.
- Handling missing data: The algorithm can handle missing values and assist in identifying categorical and numerical data.
- Flexibility: XGBoost allows customized optimization targets and split evaluation parameters that can be adapted to a wide range of problems, beyond standard regression and classification tasks.

The model shows high accuracy rates, from 0.854 to 0.877. However, further analysis using the balanced accuracy metric, to assess performance with unbalanced classes, reveals lower values ranging from 0.51 to 0.53. Although this disparity indicates that the model is effective in classifying a large number of instances correctly, it struggles to accurately represent and predict both classes equally.

B. Catboost - Roni Weinfeld

The CatBoost Classifier, a gradient boosting library tailored for efficient handling of categorical features, emerges as a notable choice. It distinguishes itself by its automatic treatment of categorical data, eliminating the need for preprocessing or one-hot encoding. Consequently, it represents one of the most promising models for our dataset. The training set er used for the CatBoost Classifier model involved minimal intervention, comprising essential feature selection and standard scaling. However, its computational demands restricted our exploration of its full potential. The default hyperparameters fitting on cross-validation resulted in train mean balanced accuracy : 0.6874 and test mean balanced accuracy : 0.6226 Following the initial implementation, we applied GridSearchCV sequentially to optimize the under and over sampling rate, as well as the n_estimators parameter. Subsequently, we conducted hyperparameter tuning, progressively adjusting parameters as computational resources permitted. These included the following parameters: number of iterations, which determines the number of boosting iterations; auto class weight, which automatically adjusts class weights to balance the dataset; tree depth, which controls the maximum depth of each tree in the ensemble; learning rate, which scales the contribution of each tree; and bagging temperature, which regulates the degree of randomness in the feature and sample selection process. Notably, the most promising parameter values identified were:

- under sample rate: 40,000
- Number of iterations: 1,000
- Auto class weight: 'Balanced'
- Tree depth: 6
- Learning rate: 0.08
- Bagging temperature: 0.4

Following the evaluation on the real test set, the obtained results exhibited a substantially lower balanced accuracy score of 0.553. This discrepancy may stem from several factors. Firstly, the challenge in running the model and optimizing hyperparameters using Optuna due to the significant time required per run, sometimes extending to a day. Secondly, despite extensive hyperparameter tuning efforts, it's plausible that the default parameters provided by the model are indeed the most suitable for this particular dataset. These

challenges hindered our ability to achieve better performance despite our best efforts in parameter optimization.

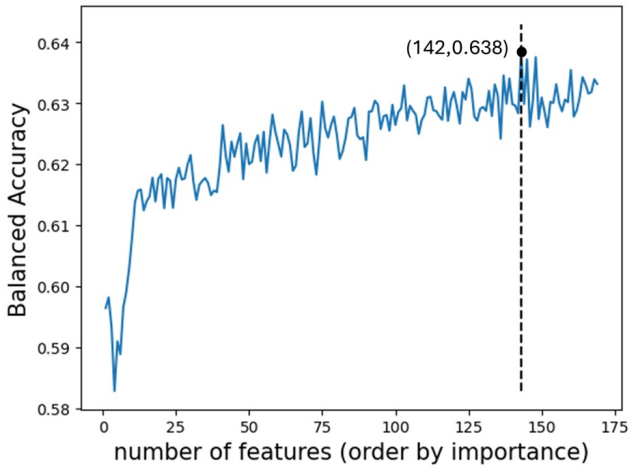

FIG. 7: The balanced accuracy as a function of the number of features. This graph was plotted based on the test dataset, after we found the final hyper-parameters based on the training and validation sets. To prevent biasing the model and leakage of data from the test set, we didn't change the model at all after observing this behaviour of the balanced accuracy on the test set.

C. LightGBM - Shachaf Frenkel

Light Gradient Boosting Machine classifier is a machine learning framework which is based on decision trees. LightGBM uses the leaf-wise tree growth algorithm. Compared with depth-wise growth, the leaf-wise algorithm can converge much faster. However, the leaf-wise growth may be over-fitting if not used with the appropriate parameters. Therefore, our main effort will be focused around preventing over-fitting. Like CatBoost, LightGBM can handle missing data and categorical features (without OHE, with integer-encoded categorical features)- LightGBM applies Fisher's method [10] to find the optimal split over categories. The default hyperparameters (with $49K$ samples from the "other" class and 97 samples from the "< 30" class) with un_balanced=True and categorical features coding with Fisher) resulted in train mean balanced accuracy : 0.6977 and test mean balanced accuracy : 0.6269. We implemnted GridSearchCV and for search over the num_leaves, we found that the optimal leaves number is 10 with:balanced accuracy - num_leaves 0.62835 on the validation set. For grid search over both num_leaves and max_depth: max_depth: 15, num_leaves: 25 and the balanced accuracy result - 0.6290903292187562. The lightGBM balanced accuracy on the test set was 0.6288. The hyper-parameters we tuned were:

- Number of leaves: max number of leaves in one tree. This is the main parameter to control the complexity of the tree model.
- Maximal depth: controls the maximum distance between the root node of each tree and a leaf node.
- Boosting type: there are different types of Gradient boosting methods- gdbt, DART and GOSS. We found that DART out-perform the other methods in out task [13].
- Number of iterations: specifies the number of boosting iterations (trees to build).
- Learning rate: also known as shrinkage, scales the contribution of each new tree to the ensemble.
- Minimum data in a leaf: as it grows, it prevents over-fitting in a leaf-wise tree but might as well cause under-fitting when too large (bias-variance tradeoff).

In addition, we set "is_unbalanced" to be True to give extra weight to the minority class. We also set a few hyper-parameters that should help fight over-fitting: bagging (frequency and fraction), min_split_gain, cat_l2, reg_alpha and reg_lambda. All parameters were tuned with a grid search, using the GridSearchCV of sklearn and Optuna as discussed later. Eventually, the non-default regularization terms didn't yield a good enough result so the final hyperparameters we used were only a very few: $categorical\_feature = indices$,$objective =$ "$binary$",$is\_unbalance = True$,$num\_leaves = 25$,$max\_depth = 15$ and $boosting\_type = dart$.

D. RandomForest classifier - Maya Sheffer

The random forest algorithm, proposed by L. Breiman [14], has been extremely successful as a general-purpose classification method. The approach, which combines several randomized decision trees and aggregates their predictions by averaging, has shown robust performance over various datasets. We used a **balanced random forest**, which differs from a classical random forest by the fact that it will draw a bootstrap sample from the minority class and sample with replacement the same number of samples from the majority class. First, we evaluate the primary results using default parameters and without performing over or under-sampling of the data set. Results with default parameters - train mean accuracy: 0.8603, test mean accuracy: 0.6169. Second, we optimize the most significant hyper-parameters using a basic grid search and cross-validation over 10 folds. The first parameter we

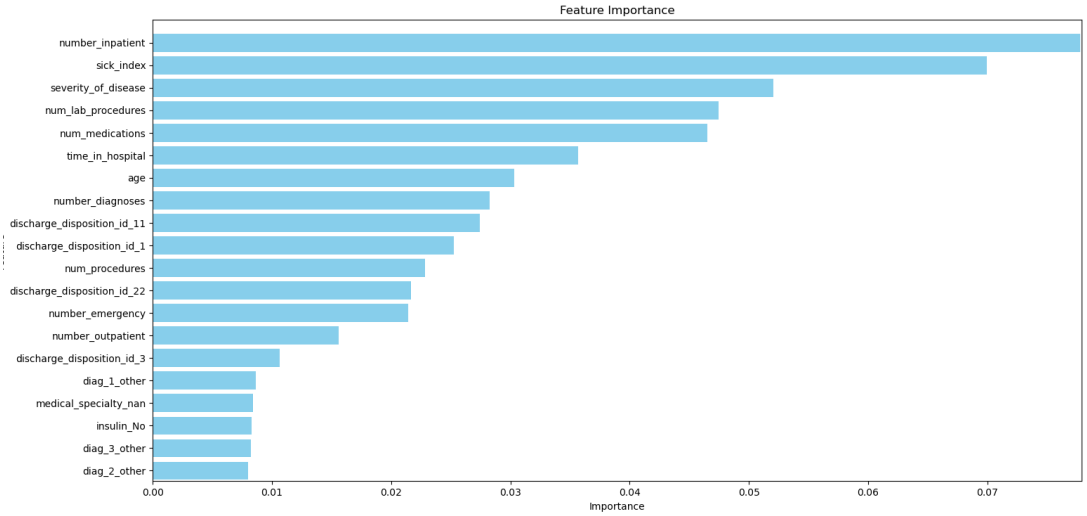

FIG. 8: Feature importance of an example model shows the relevance of the new features engineered by us- sick index and severity of the disease.

optimized was max depth, we set the range of the search to be 3-100 and found that a depth of 16 gave the best mean balanced-accuracy score (0.6256). The second one was n_estimators. We optimize this parameter using the max depth of 16. The best results were of 114 estimators, from a range of 5 to 200 (0.6259). Then, we fine-tuned these parameters and tuned the rest of the model parameters, using the Optuna framework. In this step, we used the values of the range around the parameters results from the first hyper-parameter tuning and added the undersample ratio as a hyper-parameter. 'under_samp': 10554, 'max_depth': 16, 'n_estimators': 114 Balanced accuracy: 0.626655. Then we used OPTUNA to tune 'criterion', 'min_samples_split', 'min_samples_split', 'min_samples_leaf', 'bootstrap and, 'class_weight' parameters over 800 trials 6. The chosen parameters :

- **criterion** : entropy
- **min samples split**: 3
- **min samples leaf**: 1
- **bootstrap** : True
- **class weight** : balanced subsample

Balanced accuracy: 0.629807. Finally, we evaluate the model results over the test set. We trained an imbalanced random forest over all the training data and used the parameters we found in the hyper-parameters tuning stage. We preprocessed the test set using the fitted pipeline used on the train set and predicted the labels over the test dataset features. The evaluation was over 15 seeds. Results - test mean balanced accuracy: 0.6317 (see full results in the supplementary files).

	Balanced accuracy score (train)	Balanced accuracy score (test)
Random forest	0.7306	0.6
LGBM	0.6607	0.6274
XG boost	0.90908	0.5337
Cat Boost	0.6874	0.6226

FIG. 9: The balanced accuracy results for each model on the training set and the validation set using the designated CV.

	Balanced accuracy score	Precision	Recall	F1 score
Random forest	0.6317	0.1750	0.6298	0.2739
LGBM	0.6288	0.1765	0.608	0.2734
XG boost	0.52	0.25	0.06	0.1
Cat Boost	0.553	0.2	0.13	0.16

FIG. 10: The results for each model on the test set.

IV. DISCUSSION

In this project, we tried to predict early readmission in hospital of diabetic patients. Early readmission is

defined as less than 30 days. In order to do so, we used the Diabetes 130-US Hospitals for Years 1999 – 2008 from the UC Irvine ML Repository. Since this dataset contains electronic health record (EHR), it is not necessarily meant for algorithms learning it's data, many features were categorical, were in string format, had NaN values or error in values.

Our main effort was pre-processing the dataset so the different models can learn it's patterns and predict early readmission on the test set. We added a few features which turned up to be beneficial - they were ranked high in the feature importance ranking. In addition, we manually changed many features, which improved all our models' results (as we compared their results to the unprocessed dataset) While analyzing our data, we noticed that our labels are imbalanced - only 10% of the rows belonged to the minority class - which is the class we are most interested in, which is the label "< 30. Therefore, we used several method to balance our dataset - undersampling the minority class and oversampling the majority class. Under sampling the majority class was slightly improved the models results. However, oversampling didn't improve our models. We think it might be because: i. Both SMOTE and GAN detected patterns in the minority class which are the same patterns as those in the majority class - which made the synthetic data more similar to the majority class than the minority - the opposite of what we intended. ii. All our models can handle imbalanced dataset by tunning several hyperparameters. We think the models ability to handle imbalanced dataset is better than the ability of the oversampling methods to learn the minority sample - given that it has not a large number of samples. For example, neural networks such as GAN performs better on larger datasets.

We found that all model performed similarly but the random forest classifier is slightly better. This might be a result of the generality provided by the random ensemble of trees.

A major lesson we learned while working on our project is the importance of objective and neutral validation set, that will be a similar as possible to the test set. We made an effort to over sample and undersample only the train set and to split the train and validation sets such that patients that appear several times in the dataset will always be in the same set together. This significantly changed our validation set balanced accuracy result, which obviously was much higher when the same patient appeared in both train and validation set. In addition, we learned of the importance of choosing a good performance measure. When we naively tried to run the models and evaluate them on the accuracy score, we got a very high result. This only represents that the dataset is unbalanced and that the model almost always predicts the majority class and therefore is "correct". After choosing balanced accuracy performance measure, we could truly evaluate the model.

[1] C. K. D. J. Clore, John and B. Strack, Diabetes 130-US Hospitals for Years 1999-2008, UCI Machine Learning Repository (2014), DOI: https://doi.org/10.24432/C5230J.

[2] Roglic Gojka, Who Global report on diabetes: A summary, International Journal of Noncommunicable Diseases **1**, 3 (2016).

[3] Samah Alajmani and Hanan Elazhary, Hospital readmission prediction using machine learning techniques, International Journal of Advanced Computer Science and Applications **10** (2019).

[4] Rohit R Dixit, Risk assessment for hospital readmissions: Insights from machine learning algorithms, Sage Science Review of Applied Machine Learning **4** (2021).

[5] Manna S. and Malathi G., Performance analysis of classification algorithm on diabetes healthcare dataset, International Journal of Research-Granthaalayah. **5** (2017).

[6] Beata Stracki et al., Impact of hba1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records, BioMed Research International (2014).

[7] Dhiraj Kumar Sah Kanu, Implementation of big data analytics on diabetes 130-us hospitals for year 1999-2008 for predicting patient readmission, DOI:10.13140/RG.2.2.18564.30081.

[8] Spelmen and Porkodi, A review on handling imbalanced data, international conference on current trends towards converging technologies (2018).

[9] Ramyachitra and Manikandan, Imbalanced dataset classification and solutions: a review, International Journal of Computing and Business Research **5**, 1 (2014).

[10] Walter D. Fisher, On grouping for maximum homogeneity, Journal of the American Statistical Association **53** (1958).

[11] Chawla, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research **16** (2002).

[12] sdv team, *Demystifying the CTGAN Loss Function* (2022).

[13] K.V. Rashmi and Ran Gilad Bachrach, Dropouts meet multiple additive regression trees, Proceedings of the 18th International Conference on Artificial Intelligence and Statistics **38** (2015).

[14] K.V. Rashmi and Ran Gilad Bachrach, A random forest guided tour, Test **25** (2015).