READY TO BE DISCHARGED:

EXAMINING HOSPITAL READMISSIONS

Machine Learning Project

André Silva (20230972@novaims.unl.pt)
João Martins (r20201567@novaims.unl.pt)
Leon Debatin (20230549@novaims.unl.pt)
Michelle Buston Vega (20230590@novaims.unl.pt)
Raquel Mendes (20230596@novaims.unl.pt)

ABSTRACT

Patient readmission into hospitals is an undesirable outcome, sometimes unavoidable, but that can many times be avoidable, by providing proper care and personalizing treatment to individual needs. Given that different studies show that hospitals do not have much better tools to guess patient readmissions than random guessing, in this project we dedicate ourselves into employing machine learning models to try and predict hospital readmissions, first on a binary Yes/No fashion, and then on a Multiclass No/<30 days/>30 days fashion. We believe that, with proper use of machine learning tools, costs to national health systems can be reduced, and health outcomes can be improved.

We employ a variety of methods: Logistic Regression, Random Forest, MLP, HGB, SVM, KNN, GaussianNB, among others. Computational power is also a limitation in our work.

Our findings show that this problem is, indeed, complicated. We obtain F1 Scores of 0.373 for the binary prediction problem, and 0.631 F1 weighted scores for the multiclass prediction problem. While we believe our results don't have a large margin to be improved on, we believe that better data will yield better results in the future. We also believe that collaboration with subject matter experts is the appropriate follow-up to obtain better predictions.

INTRODUCTION

Hospital readmissions represent a main challenge in the healthcare sector, bearing serious consequences for both patients and the hospitals. Multiple studies have demonstrated that high rates of readmission correlate with higher healthcare costs, decreased quality of care, and higher mortality rates in the hospital setting[1,2]. The prevalence of diabetes among hospitalized patients significantly increases the associated healthcare costs because they are more likely to be readmitted. Although various factors contribute to readmissions, many of them are preventable. For example, premature discharge or inadequate medical care during the first hospitalization.[3]

Different studies show that hospitals do not yet have accurate tools to predict readmission. The methods analyzed were only slightly better than random guessing.[4] Given the increasing pressure to reduce readmission rates, predicting hospital discharges using machine learning methods seems to be a worthwhile approach.

Our goal for this report is to find a ML approach that performs better, or at least as good as the approaches in other studies in predicting: 1. Will the patient be readmitted to the hospital within 30 days of being discharged? And 2. What specific time frame can be anticipated for a patient's readmission occurrence?

Recognizing patients at high risk of readmission at an early stage is crucial. It helps to take targeted measures to improve patient care and prevent readmissions. This report outlines the data used for modeling, data preparation steps, model development, and model evaluation procedures. By using machine learning algorithms, hospitals can optimize their operations, increase efficiency, and ultimately improve their ability to provide quality care to patients.

EXPLORATION

The first step in any scientific approach to a dataset is to understand the characteristics of the data we are dealing with. With this said, we identify our dataset as having **71236** hospital visits, with 31 features characterizing it. There are no duplicate rows, so we can safely say that we have 71236 unique hospital visits. By plotting the histograms of the numerical variables, we find that most of them do not follow a normal distribution but are skewed towards the left – we're facing skewed distributions (See [skewed distributions 1]). By plotting the boxplots of said variables, we find a lot of outliers we will need to deal with later in the Preprocessing stage of our project (*number_of_medications* and *number_lab_tests* are two examples of this phenomenon). The value count plots for the categorical variables show us that the cardinality varies a lot from feature to feature. Some features, like *payer_code* for instance, have many possible categories, but most of the distribution is focused around just a few of the existing ones (See [payer_code

[1] Sousa-Pinto, B. et al., 2013, 'Hospital Readmissions in Portugal over the Last Decade', Revista Científica da Ordem dos Médicos, no. 26, pp. 711-720

[2] Ostling, S., Wyckoff, J., Ciarkowski, S. L., Pai, C. W., Choe, H. M., Bahl, V., & Gianchandani, R. (2017). The relationship between diabetes mellitus and 30-day readmission rates. Clinical diabetes and endocrinology, 3(1), 1-8.

[3] Rubin, D.J. Hospital Readmission of Patients with Diabetes. Curr Diab Rep 15, 17 (2015). https://doi.org/10.1007/s11892-015-0584-7

[4] Allaudeen, N. et al., 2011, 'Inability of Providers to Predict Unplanned Readmissions.' Journal of General Internal Medicine, no. 26, pp. 771–76.

1]). These charts also give us a first clue into missing values, as many categories can be seen as identified with a question mark, which we assume are missing values (See [weight 1] for instance). We can also instantly identify an irrelevant variable: *country*. This data set only contains data for the USA, and so this feature will not be useful for our prediction work. Following this, we turn our attention to missing values. Most variables, categorical and numerical, do not have any missing values. However, there are some with a significant amount of them: *weight* (96.85%), *payer_code* (39.59%), *medical_specialty* (49.02%), *glucose_test_result* (94.82%), and *a1c_test_result* (83.27%). Other variables containing missing values, although in a much less worrying fashion are *race*, *age*, *admission_type*, *discharge_disposition*, *admission_source*, *primary_diagnosis*, *secondary_diagnosis*, and *additional_diagnosis.* See [NA 1].

Finally, we take a look at the balance of our target variables. An imbalanced dataset can be worrisome when predicting, as it will always be biased towards predicting the most common value of the target variable(s). Bank fraud is a good example of this – there are very few crimes happening, but if we need a model to predict them, we must build around that limitation.

In our *readmitted_binary* target variable, we have 63286 *'No'* to 7950 *'Yes'*, which gives us a class imbalance ratio of about 9:1. For the *readmitted_multiclass* target variable, we have 7950 *'<30 days'*, 24881 *'>30 days´*, and 38405 *´No´*, also showing a major imbalance between the three classes. See [imbalance 1].

PREPROCESSING

The next step in our work is preprocessing the data, through handling outliers, inputting missing values, feature engineering, amongst other steps. Obviously, an equivalent set of transformations will need to be performed on the train and validation samples, as well as on the test sample. In this report we will characterize the logic of the transformations applied to the train data, noting that they are the same applied to the validation and test samples besides some exceptions.

TRAIN-TEST-SPLIT

Before doing any kind of preprocessing, we split our data into train and validation samples. In fact, we already did the split before doing the EDA to avoid any type of unwanted data leakage. We use an 80/20 split instead of the usual 70/30 due to the sheer size of our dataset.

MISSING VALUES

As we begin to deal with missing values, we take specific approaches for each feature. For the variable *age* we input the mode, which in this case is the bin of [70-80]. For the race there already exists a category '?' in which we include the missing values. For *discharge_disposition*, *medical_specialty*, *admission_source*, *admission_type*, we assign them to the *´not mapped´* category. The missing values on *glucose_test_result* and *a1c_test_result* are left as *´NA´*, since we assume that the tests haven't been taken which might be valuable information.

The point of this treatment of missing values is to treat them as information as much as we can and be agnostic as much as possible in our approach. Later, our models will make a decision on whether these variables are important or not, but, for now, we make the least assumptions about the data we possibly can.

OUTLIER HANDLING

As stated above, a lot of our numerical variables have outliers as per IQR definition. Depending on the type of outliers found, we employed different strategies. The *emergency_visits_in_previous_year* and *outpatient_visits_in_previous_year* outliers with a value above 25 seemed implausible (See [visits outliers 1]), and as such were set to the median (which was 0 for all these variables). The same strategy was applied to the outliers in *length_of_stay_in_hospital*, *number_lab_tests*, and *number_of_medications*, as we reached a similar conclusion regarding the provenance of the outliers. For *number_diagnoses* and *non_lab_procedures* we decided to winsorize the data to the outer quartile range (See [number of outliers 1]), on the basis that the outliers are plausible to exist but are not useful for constructing a model that will be able to generalize the overall sample. See [outlier examples 1].

FEATURE ENGINEERING

We created new variables to try and extract more information from our dataset. We divided *number_of_medications* by *length_of_stay_in_hospital* to create *medication_per_day*. A similar process was followed to create *lab_tests_per_day*. We also decided to try to extract information from the *patient_id* variable.

As we saw in the EDA, the inpatient_visits_in_previous_year seems to correlate the most with our target variable (See [inpatient visits 1]), which is why we decided to take the mean for this variable for each patient and save it in a new feature, *mean_of_inpatient_visits*. For the validation data we create the feature by applying a left join to avoid data leakage and fill the missing values with inpatient_visits_in_previous_year. We debated a lot about why this specific variable would be so important towards predicting our target variable. It is true for many scientific subjects (Economics is a prime-example) that the best predictor of a future event, is usually if that event has happened in the past. Meaning if, on average, someone has been admitted to the hospital more times, they're likelier to be readmitted again. This is the principle we settled on, and it also has the advantage of following Occam's Razor.

FEATURE TRANSFORMATION

After some thought, we do a logarithmic transformation of the variables *outpatient_visits_in_previous_year*, *inpatient_vists_in_previous_year, and emergency_visits_in_previous_year*, as in these very left-skewed distributions it allows us to get more useful characterizations out of the variables.

For features with categories with very low frequencies, we group them into a category named "*others*". The thresholds are based on analyzing the value counts for each feature, usually it's 0.5% of the sample size (e.g. *discharge_disposition, medical_specialty*). See [threshold setting 1].

For ´*gender*´ we find that there are 3 rows with the Unknown/Invalid value. Due to the low cardinality, we set them to the mode (*Female*).

There is a very large number of unique medications in the *medication* feature. We used the International Classification of Diseases 9 (ICD9)[5] in order to encode all the primary, secondary, and additional diagnosis into categories, greatly reducing dimensionality within the features.

Finally, we decided to apply frequency encoding to every feature with more than two categories, to avoid a sparse dataset. Only for the binary categories did we use one-hot-encoding. Through frequency encoding we might risk some loss of information compared to one-hot encoding every variable but given our data's characteristics this is a decision we are comfortable with. The idea behind frequency encoding is to assign to each category of a categorical variable its frequency, so that we convert it from a categorical variable to a numerical one. Frequency encoding is also useful in the sense that we can run feature selection together for all variables, instead of using separate methods for numerical and categorical, leading to efficiency gains and possibly uncovering some synergetic effects regarding prediction that were out of our reach beforehand. The last step for our initial preprocessing is to scale the frequency encoded and numerical variables. We chose the Standard Scaler approach for our scaling.

BINARY CLASSIFICATION

FEATURE SELECTION

From the get-go we drop the *patient_id* and *encounter_id* since they are ids as well as *country* because it is a constant value: "USA". After this, we perform the feature selection together for both the numeric and categorical variables. We employed a majority vote rule using the following methods: Mutual Information, LassoCV, Mann–Whitney U Test, Recursive Feature Elimination with Logistic Regression, and Recursive Feature Elimination with Decision Trees. This way we ensure using filter, wrapper and embedded feature selection methods for feature selection.

The Mann-Whitney U-test is a non-parametric test. It is used to check whether two independent samples come from the same population. The Mann-Whitney U-test is mainly used if the data are ordinally scaled or if the conditions for the unpaired t-test are not met. The advantage of the Mann-Whitney U Test over the Wilcoxon test is that the first can compare variables of different lengths, which we have because of the class imbalance.

We choose the Mann-Whitney U Test over an ANOVA or T-test because they require normal distributed variables and our data has a lot of skewed distributions. For the RFE we start using one feature and iterate until we use all features to see which features improve the stratified 5-fold cross validation (CV) score.

We defined thresholds (thresholds majority vote) for whether to keep the variables or not for the Mutual Information and Mann-Whitney U Test, and then summed them up together for the final majority vote.

Based on these criteria, we keep the following variables: ***age, payer_code, medication, discharge_disposition, prescribed_diabetes_meds emergency_visits_in_previous_year, inpatient_visits_in_previous_year, number_of_medications, number_diagnoses, mean_of_inpatient_visits and change_in_med_during_hospitalization***. See [Majority Vote 1]

[5] https://www.cdc.gov/nchs/icd/icd9.Htm

Hereby, the feature importances of the decision tree as well as the coefficients of the logistic regression identify the *mean_of_the_inpatient_visits* as the most important feature by far, followed by *inpatient_visits_previous_year*. Also the *discharge_disposition*, *prescribed_diabetes_meds* and the *emergency_visits_in_previous_year* seem to be relevant.

See [feature importance and logreg coeffs 1].

Despite having a Spearman Correlation of 0.9 between *mean_of_inpatient_visits* and *inpatient_visits_in_previous_year* we decide to keep both features because the decision tree and the logistic regression have a better performance using both features which leads us to conclude there are beneficial synergies. The multicollinearity of the features is also visible having a look at the coefficients of the logistic regression, where we can see a negative sign with a high value for the *mean_of_inpatient_visits* coefficient, even though we know from the EDA it correlates positively with our target variable, which might be read as another indication of beneficial synergy for the prediction.

MODELS

After successfully completing the data pre-processing, our attention turned to the predictive modeling phase. Therefore, based on a meta-study (See [Meta study 1]) which mentions the most frequently used machine learning algorithms for hospital readmission prediction, we decided to use the following models: Logistic Regression (considering L1 and L2 regularization), Random Forest, Boosted Tree Methods, SVCs, Naive Bayes, Muli-layer Perceptron (MLP), KNN and lastly Decision Trees. For the Boosted Tree Methods we decide on using a HistGradientBoosting. This stands for Histogram-Based Gradient Boosting, and it is an ensemble learning method implementation of the gradient boosting algorithm that specifically leverages histogram-based techniques for faster and more memory-efficient training. The histogram-based approach involves discretizing the continuous input features into discrete bins and constructing histograms to represent the distribution of data within these bins. This allows for more efficient computation of the gradient during the training process.

For the linear Kernel we use the Stochastic Gradient Descent (SGD), which is an extremely efficient approach in fitting linear classifiers when we have large datasets. It is also very easy to implement, which means it garners a lot of opportunities for tuning to individual needs. We combine it with a hinge loss-function which, in essence, makes it equivalent to a linear SVM but with much more efficient processing.

STRATEGIC APPROACH

Our general strategy is to apply a grid search with a stratified 5-fold CV. We train the models with best found parameters based on the CV score to calculate the train and test error. We start with a broad parameter choice for the grid search for each algorithm. After that the parameter grids get adapted with multiple iterations based on our theoretical machine learning knowledge, trying to further optimize the F1 score for the test set each time until we cannot see significant improvements anymore. We justify the 5-fold over 10-folds due to the size of the dataset for a more efficient running time. Facing long running times we decided to renounce applying multiple repetition within the CV as well, which gives us more time to try out different parameter grids.

Because of the class imbalance we'll assess its performance using the F1 score, a metric that helps strike a balance between precision and recall. To address class imbalance, we considered two primary strategies: random under sampling (RUS) and sample weights. Our initial thoughts led us to consider including oversampling techniques such as Synthetic Minority Oversampling Technique (SMOTE) and random oversampling (ROS) as well. However, since the gap between the sample weights scores and the RUS scores was so marginal as we will see in the evaluation, we concluded that there is no significant loss of information through the under sampling process, thus leading us not to use it.

For our Kaggle solution we decided to do the preprocessing steps again for all the training data and train our best model with the best hyperparameters based on the training data from the train/test-split. The main reason is to include all available information of our most important feature *mean_of_inpatient_visits* by uncovering more duplicated patients.

RESULTS

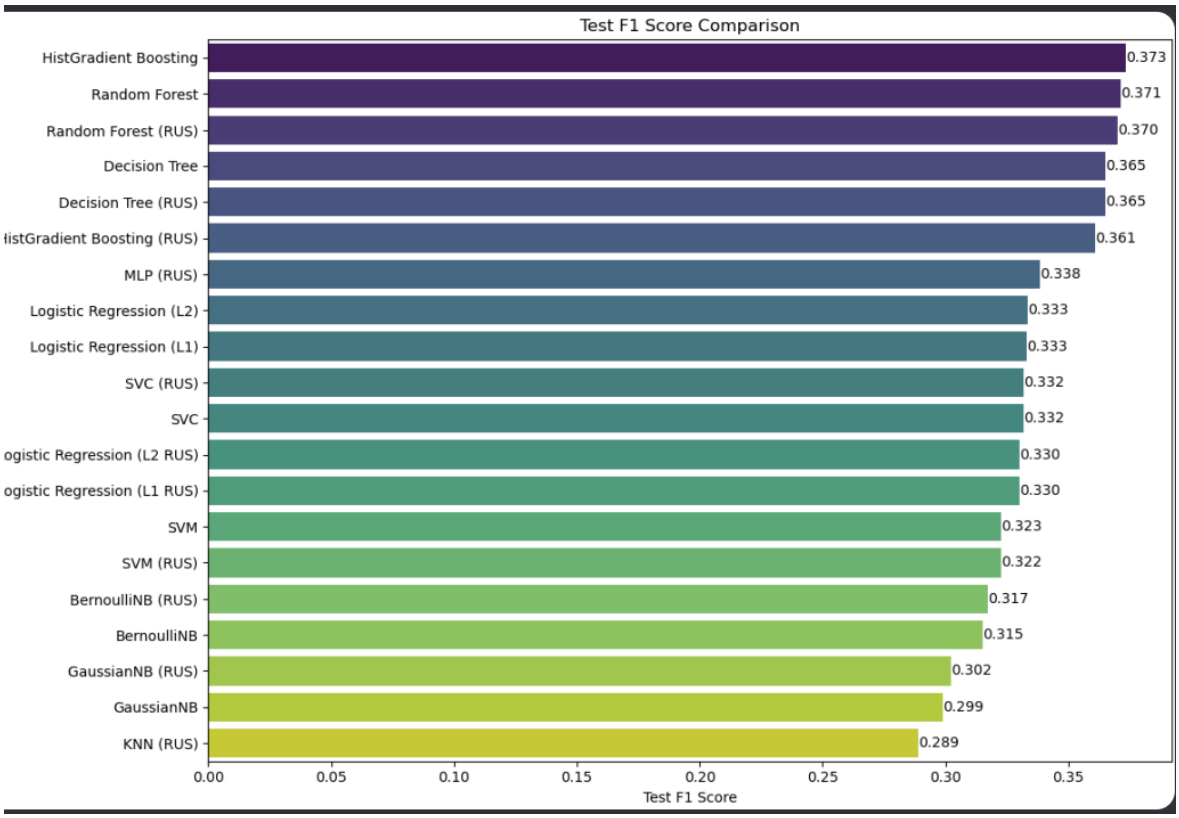


Figure 1: Binary Classification Results

Our best model is the Histgradient Boosting model with an F1 score for the test data of 0.3734 followed by the Random Forest and the Decision Tree. In general, we can observe that the Tree-based methods seem to perform the best on the dataset by far. The MLP slightly outperformed the logistic regressions and the SVC with an RBF kernel. The Naïve Bayes models have comparably low F1 scores and the KNN performed the worst with an F1 score of 0.289. In general, we can observe that the RUS approach always performs slightly but not remarkably worse than the sample weights approach.

Comparing the gap between the training and test scores we see the biggest gap for Random Forest with a F1 train score of 0.42 and a test score of 0.37, which could be a sign of overfitting.

But since it performs the second best on the test data, we have no reason to assume that there is a lack of generalization due to overfitting. See [Binary Results Table 1].

Having a look at the AUCs we can see that most of them are in a range between 0.67 and 0.7, which suits the findings of the meta study. The most remarkable difference is the performance of the ANNs analyzed in the meta study with a median AUC of 0.71. See [AUC meta study 1]. Whereas our MLP only has an AUC of 0.68. This might be due to the fact that the ANNS from the study are RNNs and CNNs which are most frequently applied for time series prediction[6], and we don't have a date feature in our data which means we have less information available.

	Readmission 1 True	Readmission 0 True
Readmission 1 Pred	0.065343	0.173077
Readmission 0 Pred	0.046252	0.715328

Figure 2: Confusion Matrix for Binary Classification Results

Having a look at the confusion matrix of the HistGradientBoosting model, we can see that we have a TP rate of 6.53%, a TN rate of 71,53%, a FN rate of 4,63% and a FP rate of 17.31%.

This leads to a recall of 0.59 and a precision of 0.27. This means only approximately 1 out of 4 predictions where we assume an upcoming readmission is correct. Still, we miss predicting approximately 4 out of 10 readmissions.

In practical application one would unnecessarily scare 3 out of 4 patients by informing them they have a higher risk of readmission or keep them longer in the hospital, while still being ignorant about 4 out of 10 upcoming readmissions. Since this might be a matter of life and death it's a tradeoff worth to take but further improving the models should still be a goal.

Coming to the Kaggle submission, following the strategy mentioned in the strategic approach, we got a training F1-score of 0.3875 when using the whole dataset and an F1-score of 0.3895 on Kaggle for the HistGradientBoosting model, which outperforms the random guessing f1-score of 0.1846 by far. By using the whole training set we can find more duplicates and adapt their *mean_of_inpatient_visits*, which leads to a significant improvement.

MULTICLASS CLASSIFICATION

ADAPTED PREPROCESSING

Our preprocessing approach for the multiclass is quite similar to the binary classification with only a few steps differing. First of all we decided to do some further feature engineering. The first feature we include is the training predictions from the binary based on our best model. Since the training error is even smaller than the test error on the Kaggle set we feel confident to say we didn't overfit our model, in which case we would have an issue regarding data leakage. Secondly, we take the mean for the emergency visits for each patient since it worked well for the inpatient visits for the binary. Here we do a left join for the validation data and fill the missing values with the *emergency_visits_in_previous_year* as well.

[6] Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J., & Chen, C. (2019). Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing*, *360*, 107-119.

Also, we had to adapt the feature selection process. The Mann-Whitney U test does not work for multiclass variables which would leave us with 4 selection methods. Since we want to do a majority vote we drop another method and decide on the mutual information criteria since we think at the end of the day it is about if a feature helps reducing the error metric and not about if some test says it is relevant, which is why we did not want to exclude any of the RFE or the L1-regularization. The majority vote of the three remaining methods leads us to the decision to keep all available and created features. This came surprisingly since the feature selection for the binary prediction suggested to drop more than half of the features, so we conclude there are a lot of features that are not relevant for the binary but are for the multiclass prediction. Regarding the logistic regression and the decision tree the most important feature is by far the *binary_prediction*. See [Feature Importance 1].

We have high Spearman Correlations between our input variables again, still we decide to only drop the *length_of_stay_in_hospital* due to high correlation with the *medications_per_day* and *lab_tests_per_day.* See [Correlation Matrix 2].

STRATEGIC APPROACH

The choice of models and the broad to small grid search approach stays the same for the multiclass. What changes is the approach of dealing with the class imbalance. After running a few models with RUS, we saw big gaps between the RUS approach and the class weights approach, which is why we decided to stick with the class weight approach only for every method that allows it, and use SMOTE for the MLP.

For the KNN we ran into an error that might have been fixed by downgrading the scikit-learn version (See [KNN Error 1]), but we didn't want to risk facing issues with the other models due to changing the scikit-learn version which is why we don't have results for the KNN for the multiclass classification.

The metric of choice for the multiclass problem is the weighted F1-score since we still face an imbalanced class, we don't assume that FN or FPs for any class are more important than for another.

RESULTS

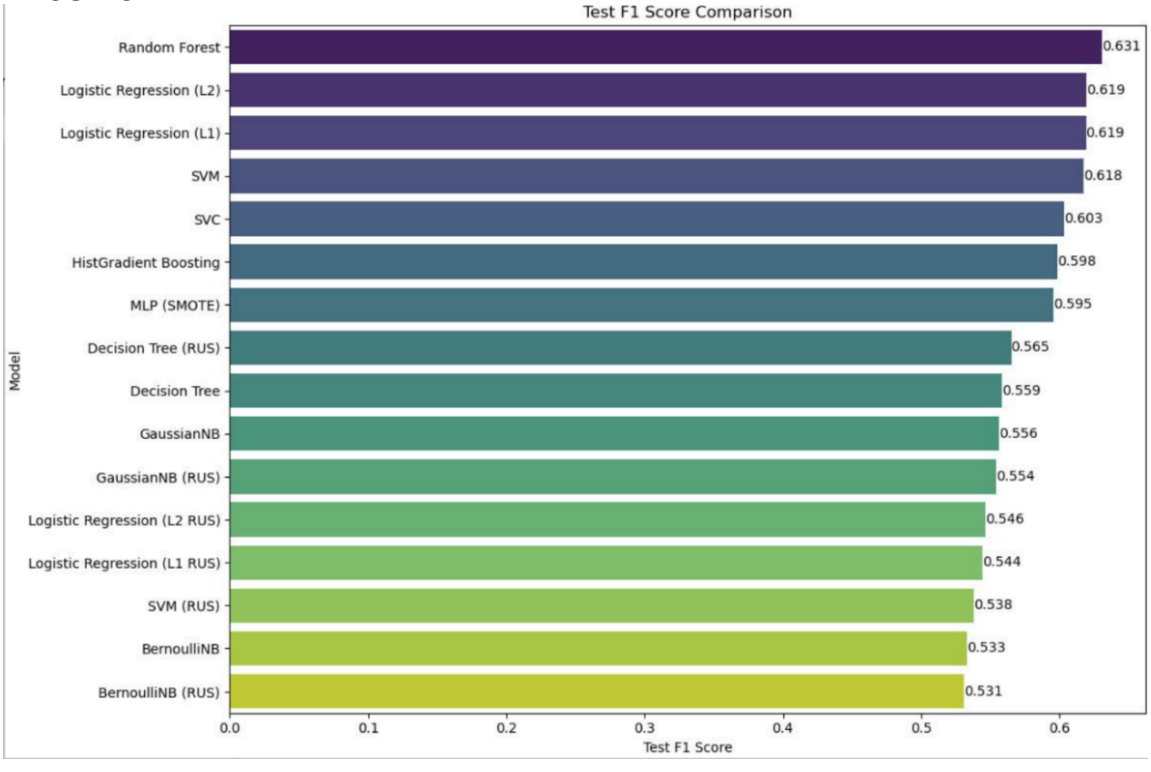


Figure 3: Multiclass Prediction Results

For the multiclass classification the best model is the random forest with a F1 score of 0.631, followed by the logistic regressions and the SVC with a linear kernel. Seemingly linear models can capture the relationship between the input variables and the multiclass readmission well.

The SVC with RBG kernel, HistGradientBoosting model and the MLP have nearly similar weighted F1 scores about 0.60. Then there is a gap of 0.03 weighted F1 scores to the other models.

This time we can see big gaps between the RUS approach for the SVCs with linear kernel and the logistic regression which indicates for a lack of information through the under sampling process.

Having a look at the differences between training and test error (See [Multiclass Results Table 1]) we can see an even bigger gap for the random forest with a weighted F1 train score of above 0.76 which again would be a sign for overfitting. We were sceptical during the hyperparameter tuning process already, when the best CV score always suggested increasing the max_depth in the next iteration, but since it's the best performing model we don't assume a lack of generalization (See [param grid rf 1]). This time we also have a big gap of train and test score for the SVC. Here as well the CV score suggested increasing the value of C, despite there was already a huge gap between train and test error (See [param grid svc 1]). Still, we come to the same conclusion as for the random forest and don't assume a lack of generalization.

For the random forest we get a weighted precision score of 0.63 and a weighted recall score of 0.64. Since those values are hard to interpret we inspect the confusion matrix. One can see that most of the readmissions <30 days are predicted as readmissions >30 days readmission and only approximately 3 out of 11 get predicted as no readmission, compared to the 4 out of 10 from the binary classification. In practical application it might be less harming to predict a < 30 days readmission as a >30 days readmission since it's still a predicted readmission, based on which one can take actions. Also worth noticing is that a lot of the readmissions > 30 days get predicted as no readmission. From the practical perspective we would assume some kind of ordinal behaviour within the multiclass which seems to be uncovered by the algorithms, since they struggle separating readmissions > 30 days from no readmissions and readmissions < 30 days, whereas readmissions < 30 days and no readmissions can be separated quite well.

	Readmission No True	Readmission >30 True	Readmission <30 True
Readmission No Pred	0.445045	0.129562	0.027442
Readmission >30 Pred	0.081836	0.157004	0.046954
Readmission <30 Pred	0.012212	0.062746	0.037198

Figure 5: Confusion Matrix for the Multiclass Predictions

CONCLUSION

SUMMARY

Following our ML process approach we ended up with HistGradientBoosting as the best model for the binary prediction with a F1-score of 0.3734 and an AUC of 0.6954 on the validation data. Comparing the AUC on the validation data of our best model to other studies we can say that our approach performs as well as most of the approaches in other studies with AUCs around 0.7. So we can conclude we didn't outperform common approaches but at least performed as good

as them and therefore reached our goal. Since we gain more information creating our feature *mean_of_inpatient_visits* using the whole training data set for preprocessing to predict the readmission for Kaggle, we reach an F1-score of 0.3894. For the Multiclass Classification the best model was a Random Forest with a weighted F1-score of 0.6303.

LIMITATIONS

The primary limitations of this work are due to a lack of time and computational power. First, there are many different preprocessing techniques that would be worth trying out, for example different methods for filling missing values such as KNN-Imputation, handling outliers in different ways like setting them to mean, median, outer quantile ranges or dropping them, as well as testing different scalers. Also, one-hot-encoding instead of frequency encoding could improve our results, since we might have lost some relevant information through frequency encoding. Furthermore, one can criticize the choice of a stratified 5-fold CV without repetitions instead of a repeated 10-fold for the grid search, which could help with the fine tuning of the models. Also, there is room for improvement regarding the feature selection. The best approach should be to do a recursive feature elimination for each model with the model itself and within the grid search, at least if the model has some kind of feature importance to enable this approach. This way you get a customized training set for each model which based on the selected feature from the model itself instead of a one size fits all training set. Another technique worth considering is building ensemble models of our best models through stacking.

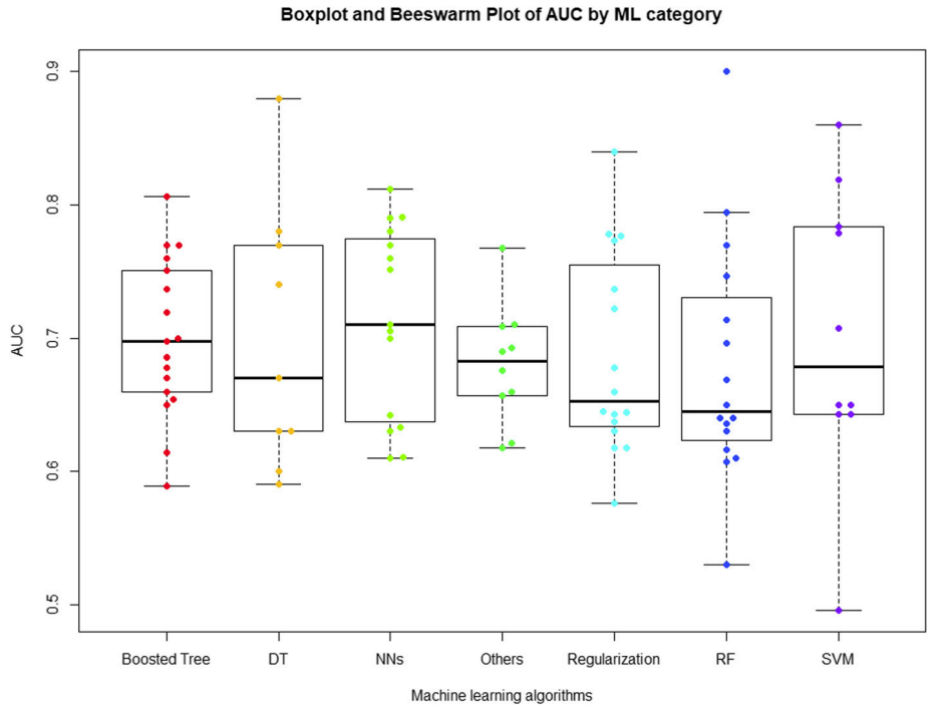
FOLLOW-UP WORK

If someone wanted to continue based on our work, we do not believe that it is possible to increase the F1-score by a lot with the given features and without any further feature engineering. We used all the most common algorithms to address the readmission classification and the algorithms can only perform as well as useful the data is: 'Garbage in, garbage out'. Therefore, we see the main issue in the available data. The limitations mentioned above might only lead to small improvements where the most relevant improvement might happen through one hot encoding and a model customized feature selection. Also collecting more data should not help a lot, at least for the binary problem, since we saw that the RUS approach leads to comparable F1-scores. We would suggest starting with an error analysis and checking if there are patterns within the false classified patients, based on which further steps can be taken. Most importantly one should contact subject matter experts who are able to provide relevant medical insights for a more appropriate preprocessing and maybe help identifying new relevant features.

REFERENCES

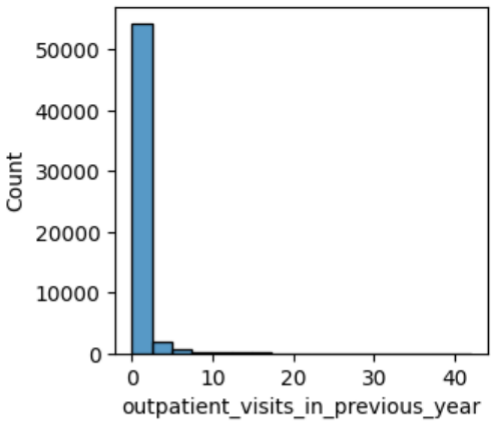
1. Sousa-Pinto, B. et al., 2013, 'Hospital Readmissions in Portugal over the Last Decade', *Revista Científica da Ordem dos Médicos*, no. 26, pp. 711-720
2. Ostling, S., Wyckoff, J., Ciarkowski, S. L., Pai, C. W., Choe, H. M., Bahl, V., & Gianchandani, R. (2017). The relationship between diabetes mellitus and 30-day readmission rates. Clinical diabetes and endocrinology, 3(1), 1-8.
3 Rubin, D.J. Hospital Readmission of Patients with Diabetes. *Curr Diab Rep* 15, 17 (2015). https://doi.org/10.1007/s11892-015-0584-7
4 Allaudeen, N. et al., 2011, 'Inability of Providers to Predict Unplanned Readmissions.' J*ournal of General Internal Medicine,* no. 26, pp. 771–76.
5 International Classification of Diseases 9, https://www.cdc.gov/nchs/icd/icd9.Htm
6 Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J., & Chen, C. (2019). Multiple convolutional neural networks for multivariate time series prediction. Neurocomputing, 360, 107-119.
7 Huang, Y., Talwar, A., Chatterjee, S., & Aparasu, R. R. (2021). Application of machine learning in predicting hospital readmissions: a scoping review of the literature. BMC medical research methodology, 21(1), 1-14.

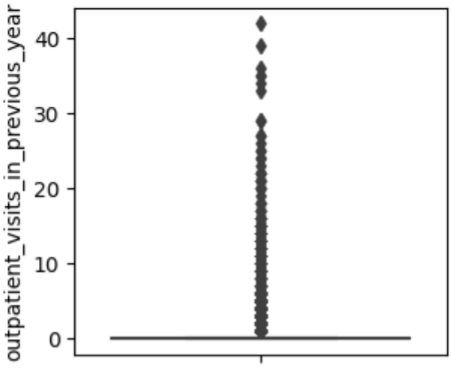
APPENDIX

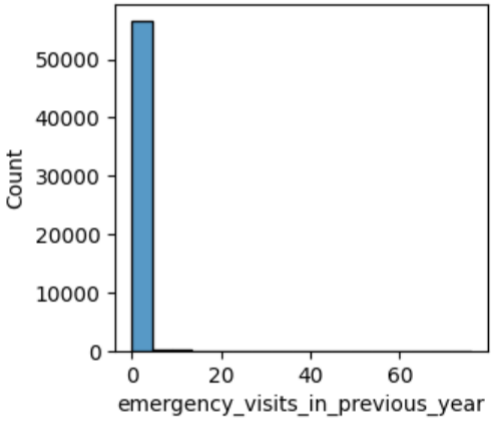


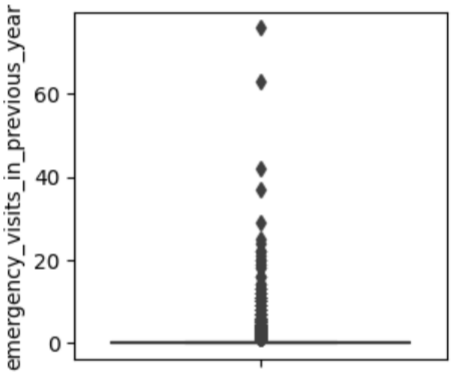
[Model Comparison from Meta Study 1]
AUC meta study 1

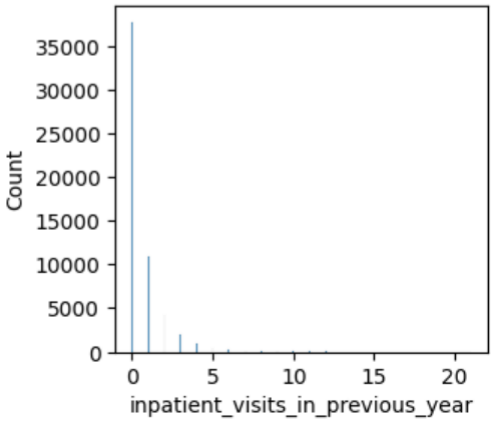
https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-021-01284-z
Meta study 1

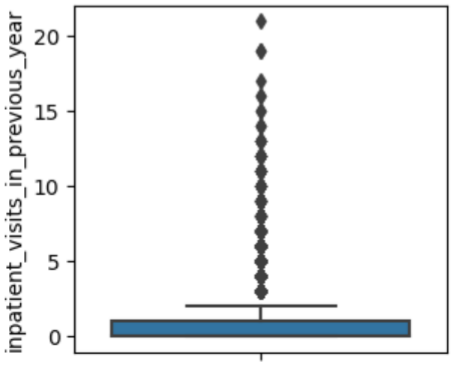

Count
50000
40000
30000
20000
10000
0
0
10
20
30
40
outpatient_visits_in_previous_year

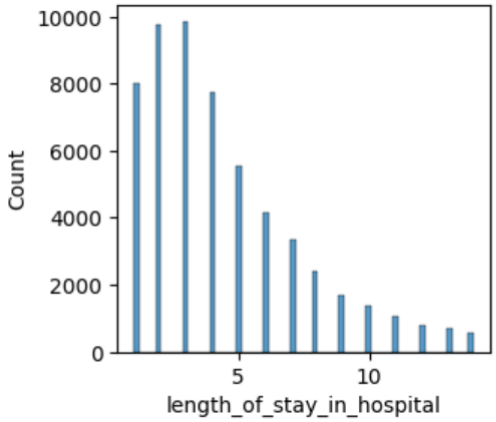

outpatient_visits_in_previous_year
40
30
20
10
0

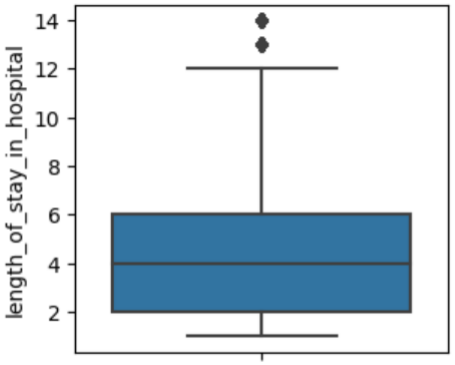

Count
50000
40000
30000
20000
10000
0
0
20
40
60
emergency_visits_in_previous_year

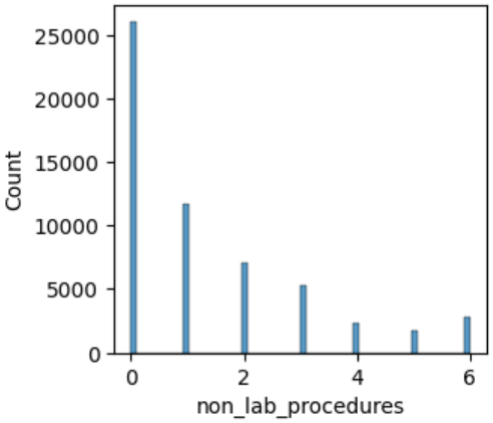

emergency_visits_in_previous_year
60
40
20
0

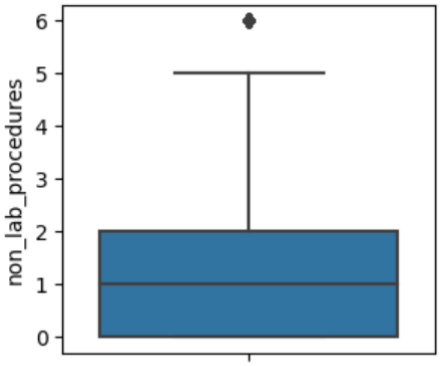

Count
35000
30000
25000
20000
15000
10000
5000
0
0
5
10
15
20
inpatient_visits_in_previous_year


inpatient_visits_in_previous_year
20
15
10
5
0



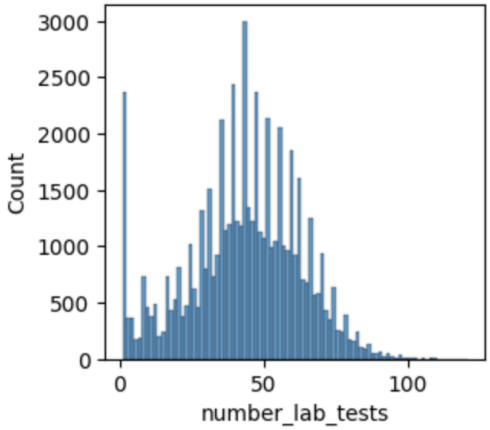


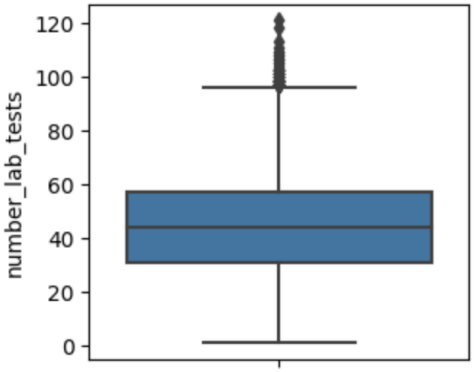


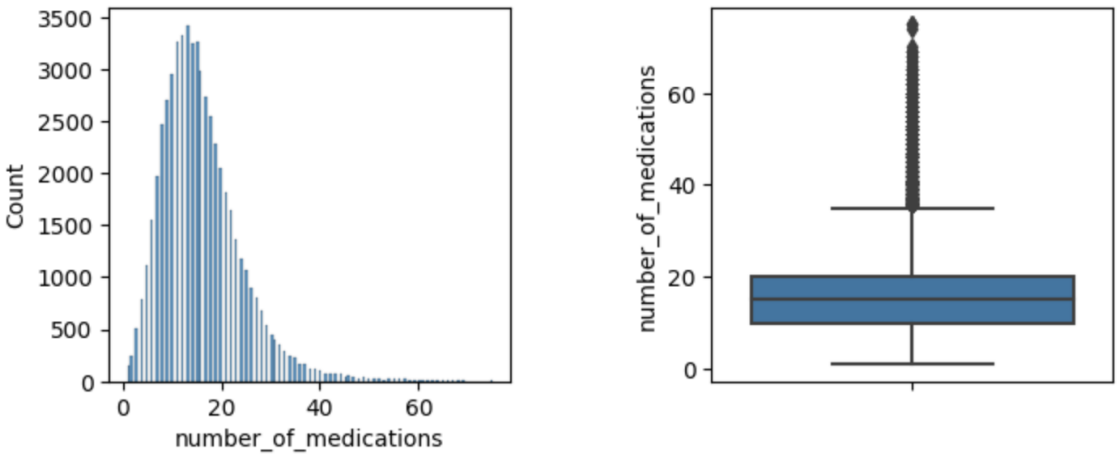


[skewed distributions]

skewed distributions 1

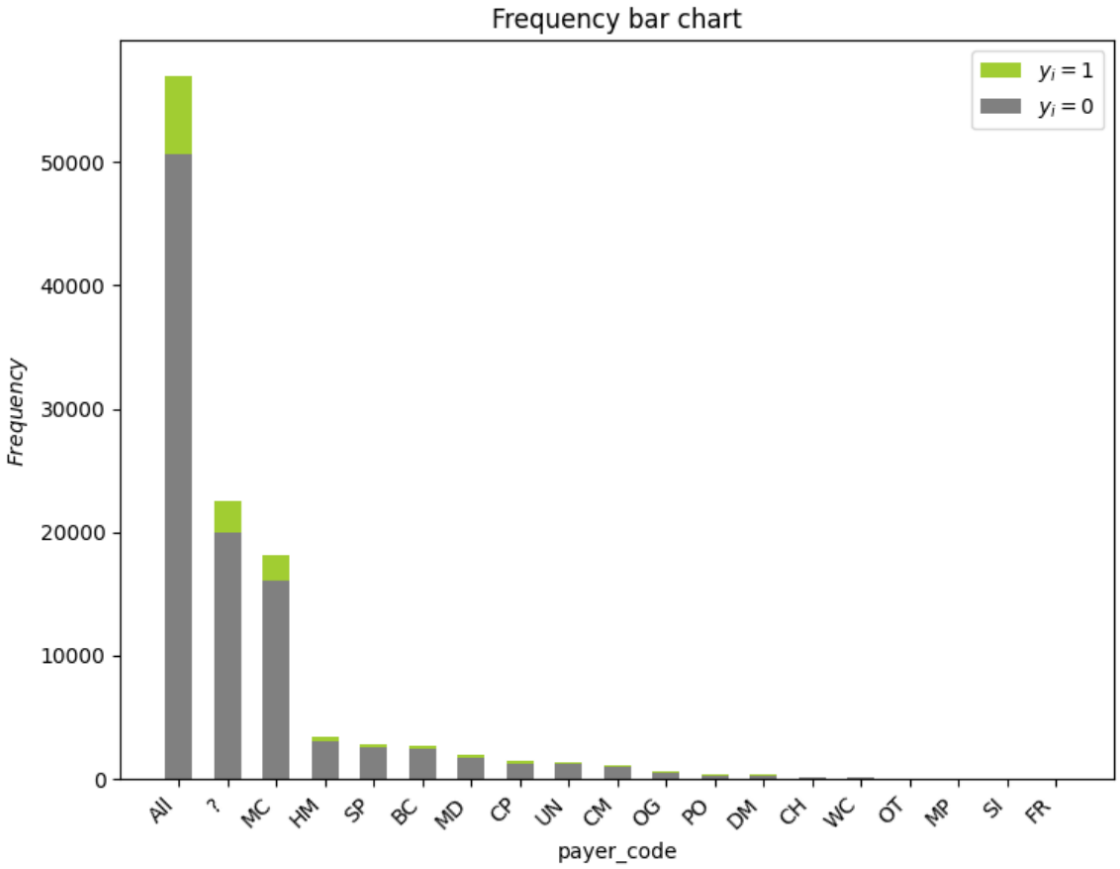






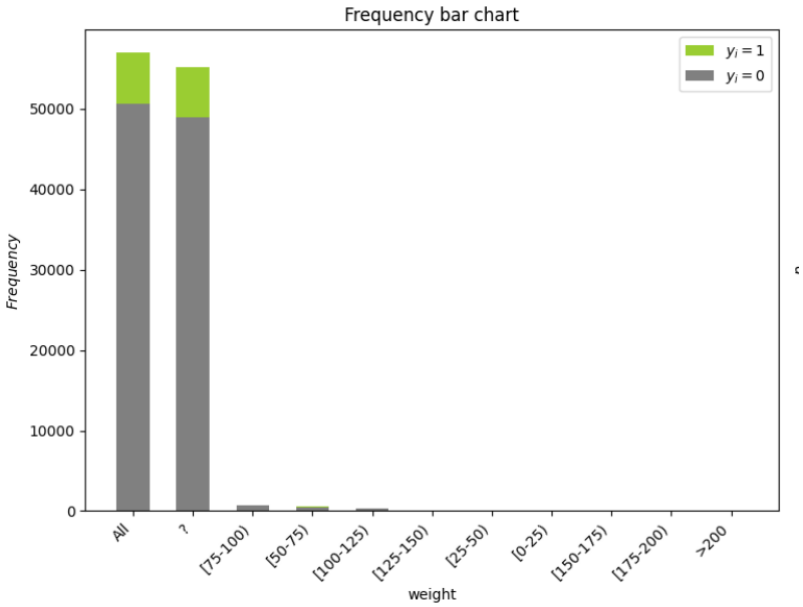
[outlier examples]

outlier examples 1



[payer_code]

payer_code 1



[weight]

weight 1

country	0
patient_id	0
race	2814
gender	0
age	2857
weight	0
payer_code	0
outpatient_visits_in_previous_year	0
emergency_visits_in_previous_year	0
inpatient_visits_in_previous_year	0
admission_type	2946
medical_specialty	0
average_pulse_bpm	0
discharge_disposition	2071
admission_source	3787
length_of_stay_in_hospital	0
number_lab_tests	0
non_lab_procedures	0
number_of_medications	0
primary_diagnosis	0
secondary_diagnosis	0
additional_diagnosis	0
number_diagnoses	0
glucose_test_result	54024
a1c_test_result	47472
change_in_meds_during_hospitalization	0
prescribed_diabetes_meds	0
medication	0

[NAs]

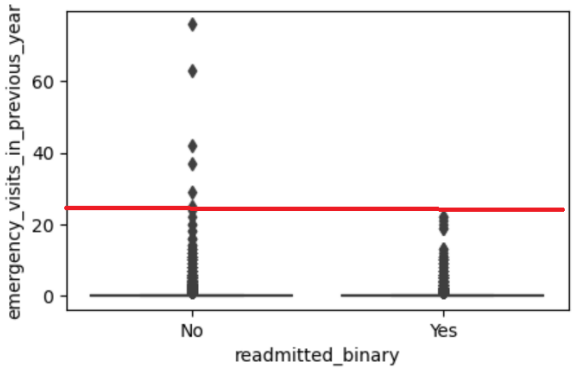
NA 1

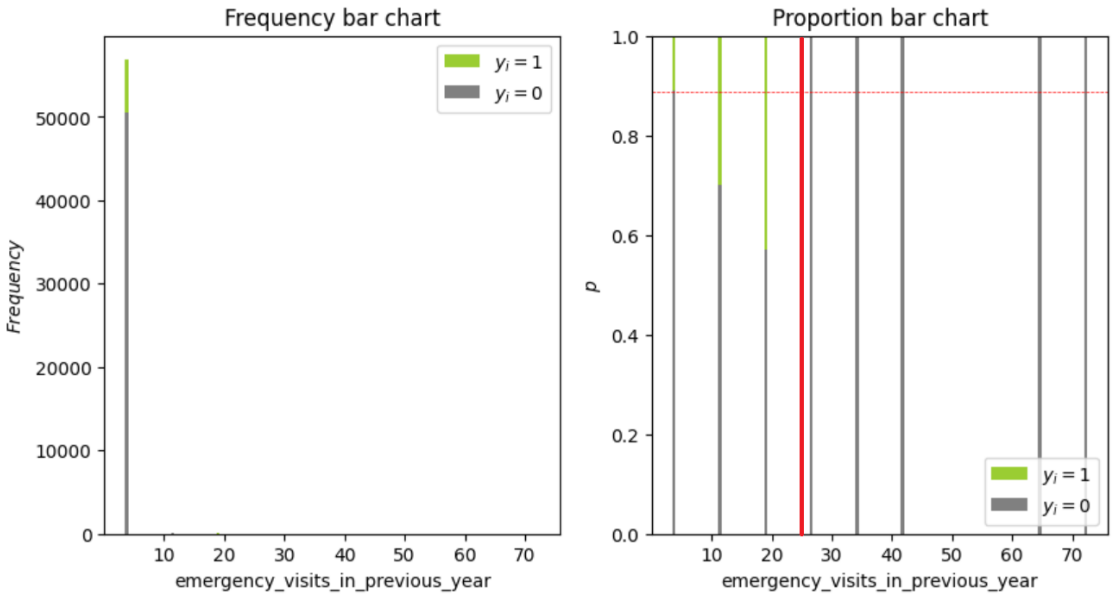
```
readmitted_binary
No     0.888399
Yes    0.111601
```

```
readmitted_multiclass
No          0.539123
>30 days    0.349276
<30 days    0.111601
```

[imbalance]

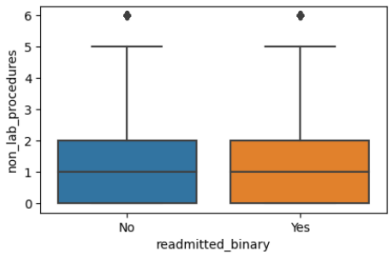
imbalance 1

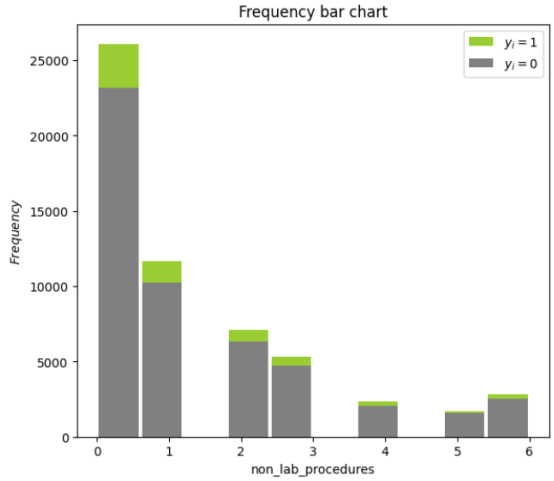


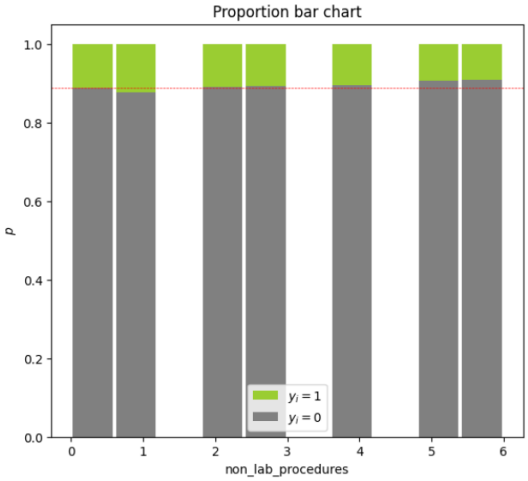


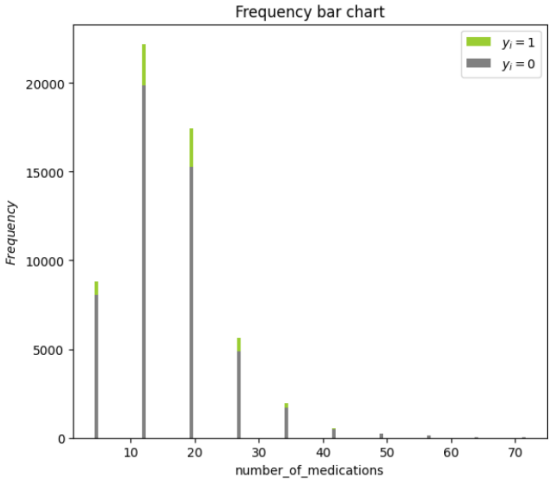
[visits outliers]

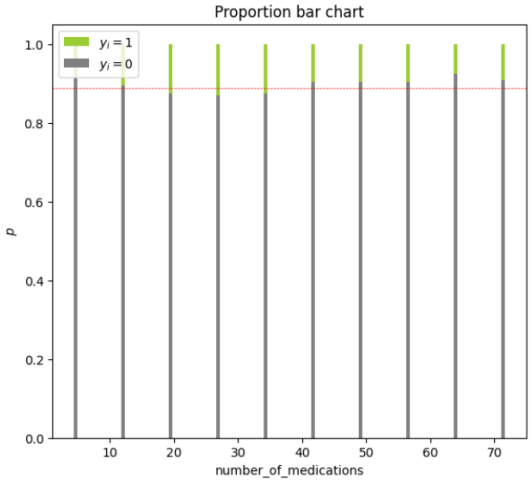
visits outliers 1

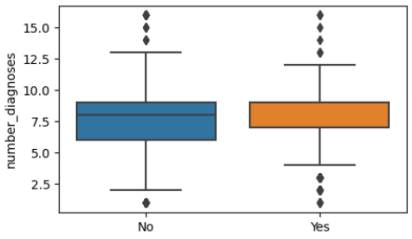




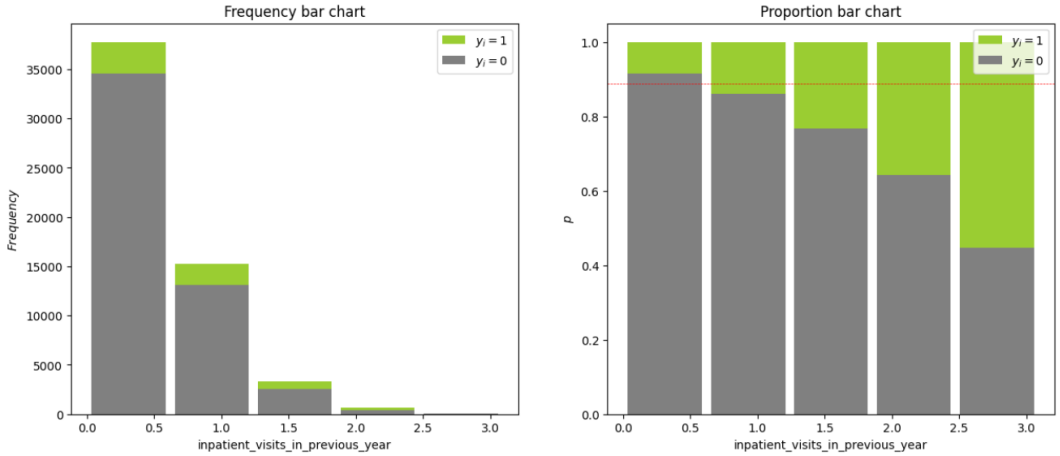








Outliers matching distributions
[nr of outliers]
number of outliers 1



Log1p of inpatient visits cal in title
[inpatient visits]
inpatient visits 1

```
feature_selection['MI_score_>_0'] = (feature_selection['MI_score'] > 0.005).astype(int)#thresholds, seems
feature_selection['Lasso_cv_>_0'] = (feature_selection['Lasso_cv'].abs() > 0.0001).astype(int)#threshold
feature_selection['Man/Chi_signif'] = (feature_selection['Mannwhitneyu/Chisquared'] < 0.05 ).astype(int)
```

[threshold majority vote]

admission_type	
Emergency	30245
Elective	10559
Urgent	10368
NA	2946
Not Available	2677
Not Mapped	176
Trauma Center	13
Newborn	4

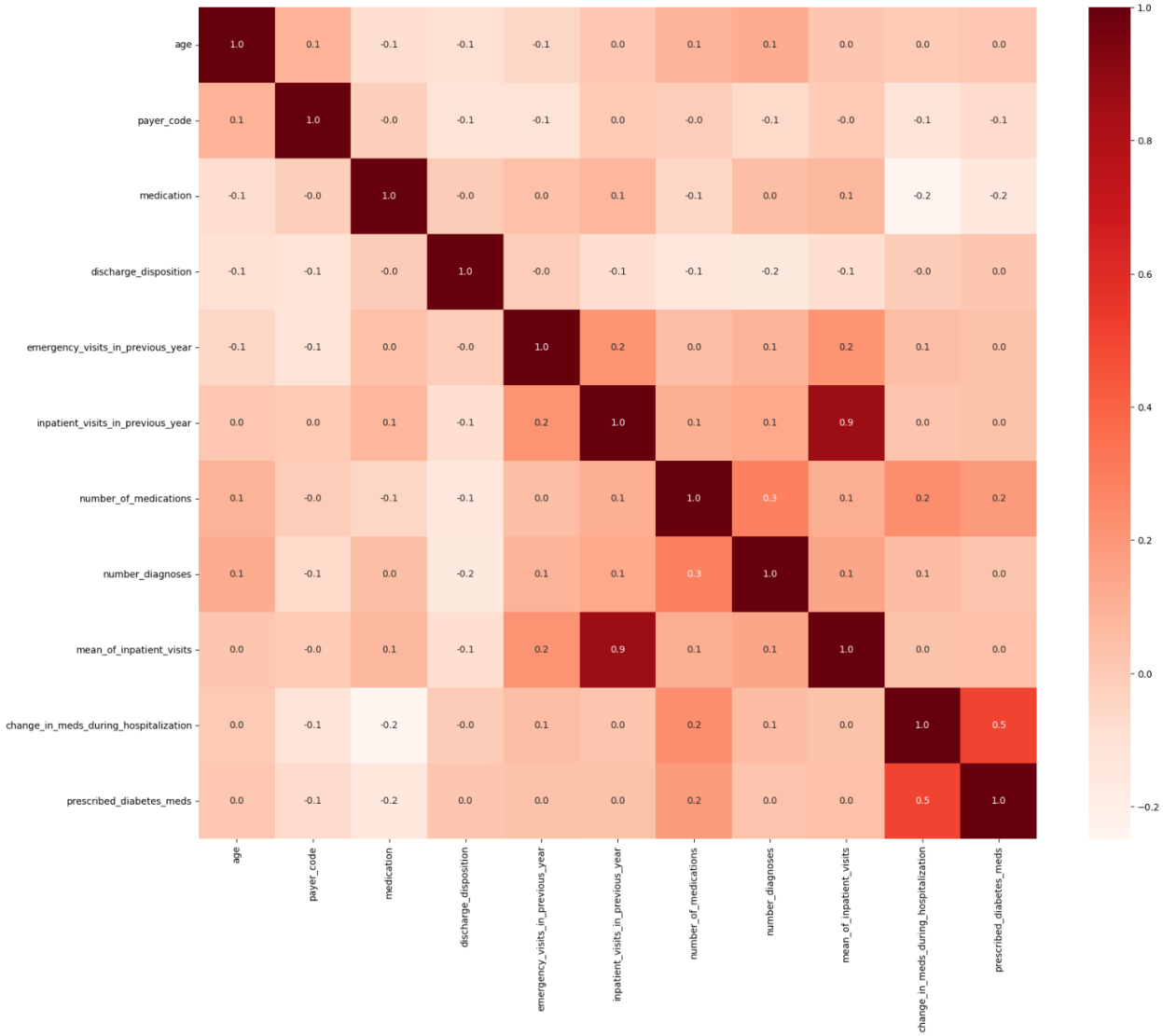
payer_code	
?	22556
MC	18176
HM	3426
SP	2839
BC	2651
MD	1986
CP	1412
UN	1374
CM	1058
OG	578
PO	321
DM	313
CH	82
WC	79
OT	55
MP	45
SI	36
FR	1

discharge_disposition	
Discharged to home	33896
Discharged/transferred to SNF	7754
Discharged/transferred to home with home health service	7221
NA	2071
Discharged/transferred to another short term hospital	1191
Discharged/transferred to another rehab fac including rehab units of a hospital .	1086
Expired	897
Discharged/transferred to another type of inpatient care institution	673
Not Mapped	543
Discharged/transferred to ICF	451
Left AMA	331
Discharged/transferred to a long term care hospital.	233
Hospice / medical facility	211
Hospice / home	196
Discharged/transferred/referred to a psychiatric hospital of psychiatric distinct part unit of a hospital	71
Discharged/transferred to home under care of Home IV provider	65
Discharged/transferred within this institution to Medicare approved swing bed	34
Discharged/transferred to a nursing facility certified under Medicaid but not certified under Medicare.	27
Admitted as an inpatient to this hospital	12
Discharged/transferred/referred another institution for outpatient services	7
Discharged/transferred/referred to this institution for outpatient services	7
Expired at home. Medicaid only, hospice.	5
Discharged/transferred to a federal health care facility.	3
Still patient or expected to return for outpatient services	2
Neonate discharged to another hospital for neonatal aftercare	1

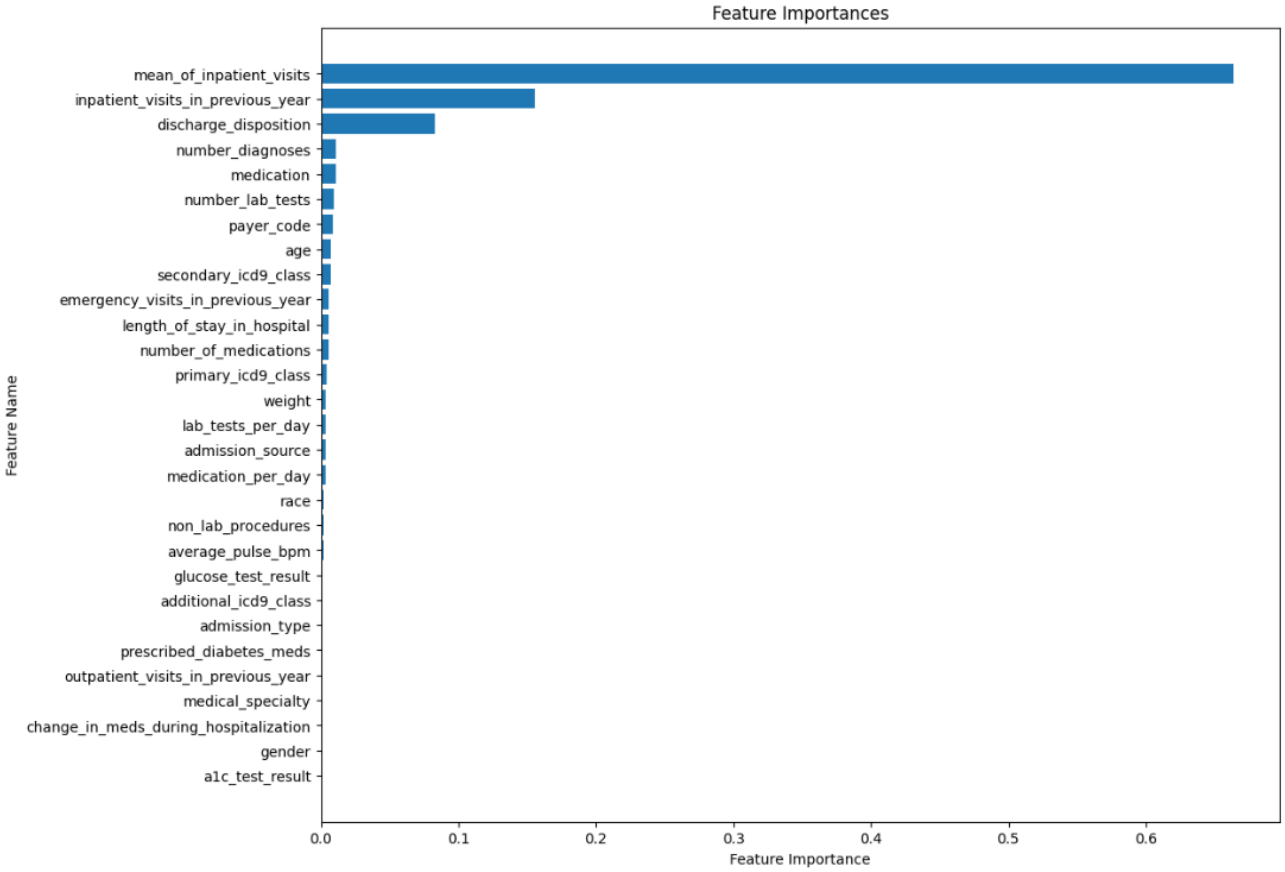
[examples for threshold setting] threshold setting 1

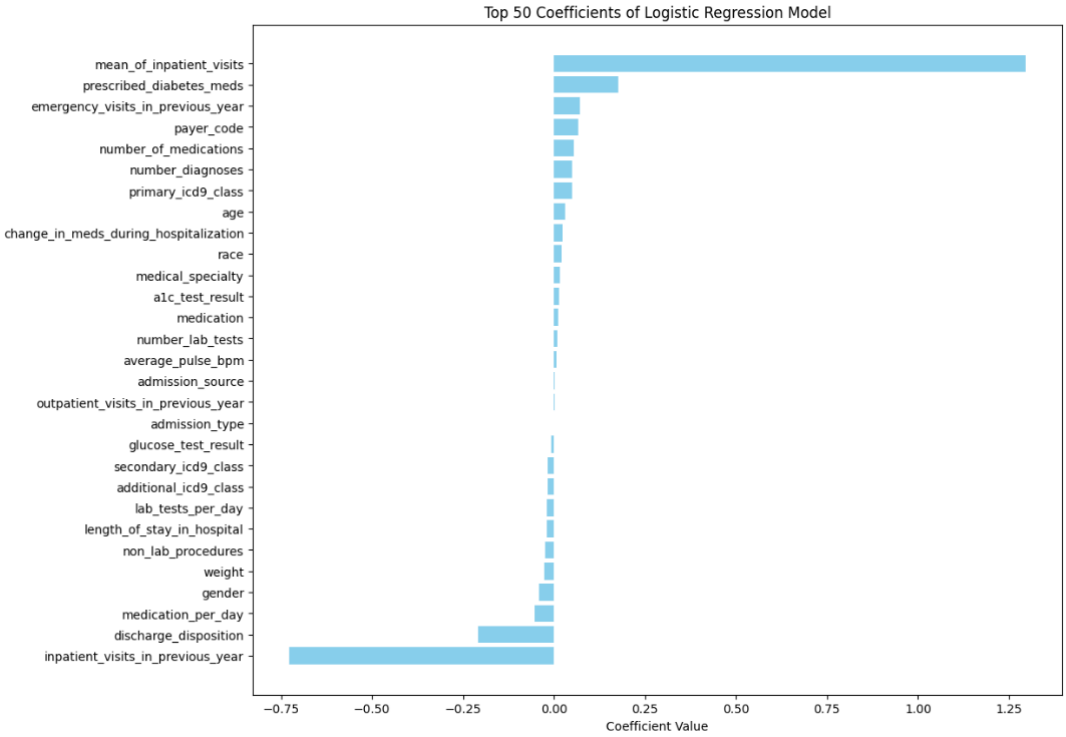
Feature	MI_score	Lasso_cv	Mannwhitneyu/Chisquared	MI_score_>_0	Lasso_cv_>_0	Man/Chi_signif	sum	RFE_log	RFE_dt
mean_of_inpatient_visits	0.040630	0.144819	0.0	1	1	1	5	1	1
inpatient_visits_in_previous_year	0.014699	-0.082103	0.0	1	1	1	5	1	1
discharge_disposition	0.008810	-0.016798	0.0	1	1	1	4	0	1
emergency_visits_in_previous_year	0.006288	0.007328	0.0	1	1	1	4	0	1
payer_code	0.004411	0.003410	0.000041	0	1	1	3	0	1
number_of_medications	0.000915	0.001865	0.0	0	1	1	3	0	1
age	0.002685	0.000131	0.010544	0	1	1	3	0	1
medication	0.003626	0.000948	0.0	0	1	1	3	0	1
number_diagnoses	0.003199	0.002386	0.0	0	1	1	3	0	1
change_in_meds_during_hospitalization	0.005429	0.000338	0.000005	1	1	1	3	0	0
prescribed_diabetes_meds	0.010611	0.008830	0.0	1	1	1	3	0	0
number_lab_tests	0.001308	-0.000000	0.000032	0	0	1	2	0	1
race	0.004407	0.001523	0.004617	0	1	1	2	0	0
lab_tests_per_day	0.000000	-0.001834	0.0	0	1	1	2	0	0
additional_icd9_class	0.003229	-0.001892	0.010914	0	1	1	2	0	0
medication_per_day	0.000000	-0.001077	0.00057	0	1	1	2	0	0
medical_specialty	0.002929	0.000885	0.000075	0	1	1	2	0	0
gender	0.008638	-0.000201	0.385694	1	1	0	2	0	0
secondary_icd9_class	0.000619	-0.000713	0.086224	0	1	0	2	0	1
length_of_stay_in_hospital	0.004510	-0.000000	0.0	0	0	1	2	0	1
outpatient_visits_in_previous_year	0.004475	-0.000000	0.0	0	0	1	1	0	0
non_lab_procedures	0.000645	-0.000000	0.031248	0	0	1	1	0	0
admission_type	0.002465	0.000000	0.046078	0	0	1	1	0	0
a1c_test_result	0.001681	0.000000	0.000401	0	0	1	1	0	0
primary_icd9_class	0.002502	0.002342	0.47877	0	1	0	1	0	0
glucose_test_result	0.003175	-0.000000	0.000808	0	0	1	1	0	0
admission_source	0.004664	-0.000000	0.000124	0	0	1	1	0	0
weight	0.001889	-0.000000	0.952231	0	0	0	0	0	0
average_pulse_bpm	0.000000	0.000000	0.722311	0	0	0	0	0	0

Sum >=3 we keep
[Majority vote]
Majority Vote 1



Correlation Matrix 1





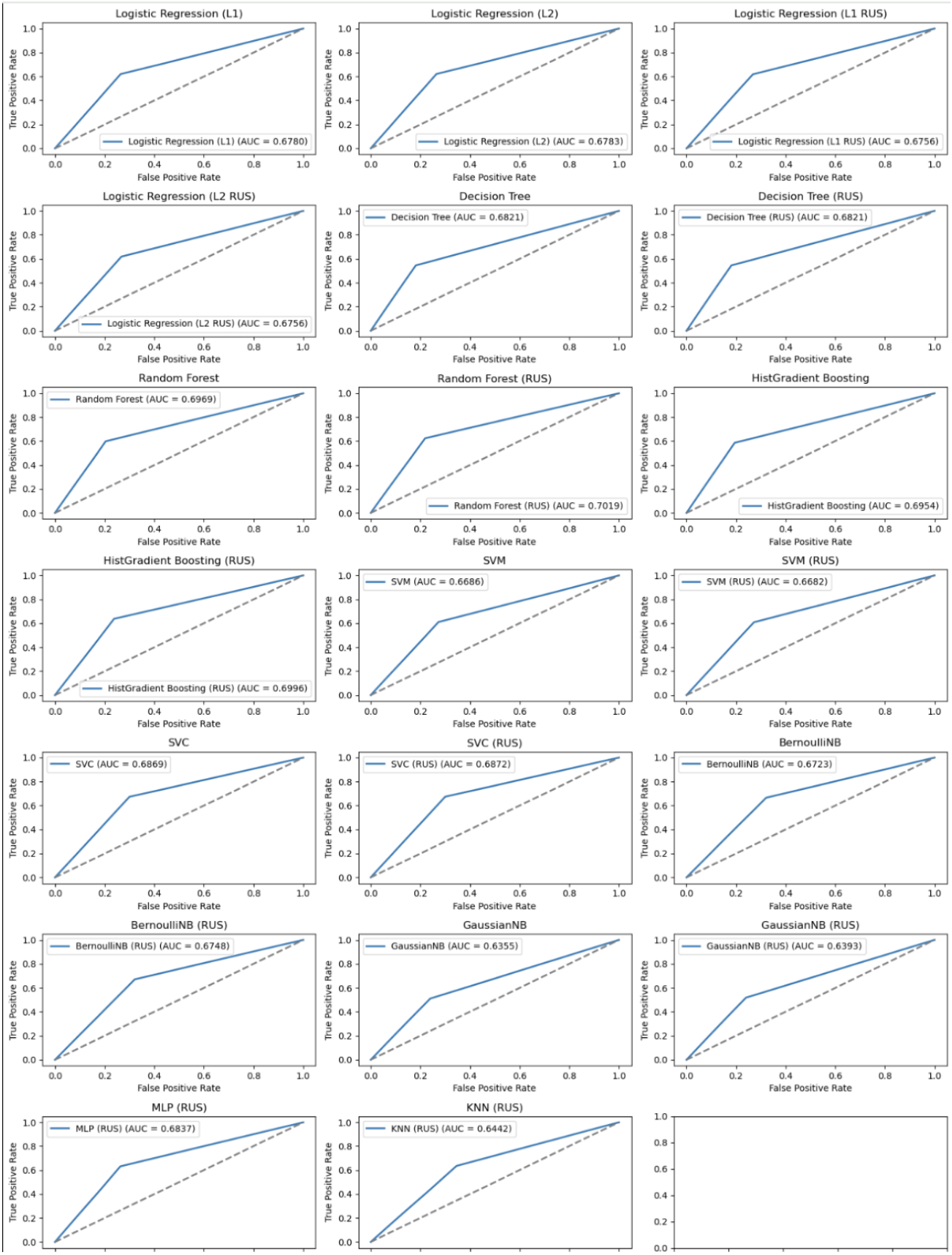
feature importance and logreg coeffs 1

	Model	Train F1 Score	Test F1 Score	CV Score
0	Logistic Regression (L1)	0.335381	0.333052	0.334995
1	Logistic Regression (L2)	0.335161	0.333277	0.335246
2	Logistic Regression (L1 RUS)	0.335235	0.330035	0.334737
3	Logistic Regression (L2 RUS)	0.335221	0.330091	0.334751
4	Decision Tree	0.367943	0.364859	0.367951
5	Decision Tree (RUS)	0.367943	0.364859	0.366697
6	Random Forest	0.415135	0.371311	0.377628
7	Random Forest (RUS)	0.367490	0.370093	0.364942
8	HistGradient Boosting	0.374188	0.373371	0.373464
9	HistGradient Boosting (RUS)	0.376409	0.360884	0.368187
10	SVC linear	0.334894	0.322527	0.333860
11	SVC linear (RUS)	0.331265	0.322441	0.333854
12	SVC rbf	0.336133	0.331632	0.330191
13	SVC rbf (RUS)	0.335252	0.331839	0.330736
14	BernoulliNB	0.296412	0.315225	0.296526
15	BernoulliNB (RUS)	0.297127	0.317276	0.295784
16	GaussianNB	0.278984	0.298804	0.278811
17	GaussianNB (RUS)	0.280076	0.302364	0.276999
18	MLP (RUS)	0.342533	0.338394	0.341993
19	KNN (RUS)	0.334637	0.289029	0.302876

Binary Results Table 1

	Model	Train F1 Score	Test F1 Score	CV Score
0	Logistic Regression (L1)	0.626450	0.619180	0.625369
1	Logistic Regression (L2)	0.626542	0.619472	0.625631
2	Logistic Regression (L1 RUS)	0.558566	0.544154	0.557675
3	Logistic Regression (L2 RUS)	0.563267	0.546265	0.563069
4	Decision Tree	0.599382	0.558556	0.579131
5	Decision Tree (RUS)	0.599510	0.565135	0.580629
6	Random Forest	0.765760	0.630846	0.646568
7	HistGradient Boosting	0.657412	0.598380	0.607717
8	SVC linear	0.617642	0.617742	0.617856
9	SVC linear (RUS)	0.540127	0.538296	0.536447
10	SVC rbf	0.727864	0.603151	0.606604
11	BernoulliNB	0.535393	0.533143	0.535630
12	BernoulliNB (RUS)	0.533021	0.530812	0.535644
13	GaussianNB	0.570610	0.556038	0.570092
14	GaussianNB (RUS)	0.568876	0.554483	0.570337
15	MLP (SMOTE)	0.611722	0.595407	0.602288

Multiclass Results Table 1

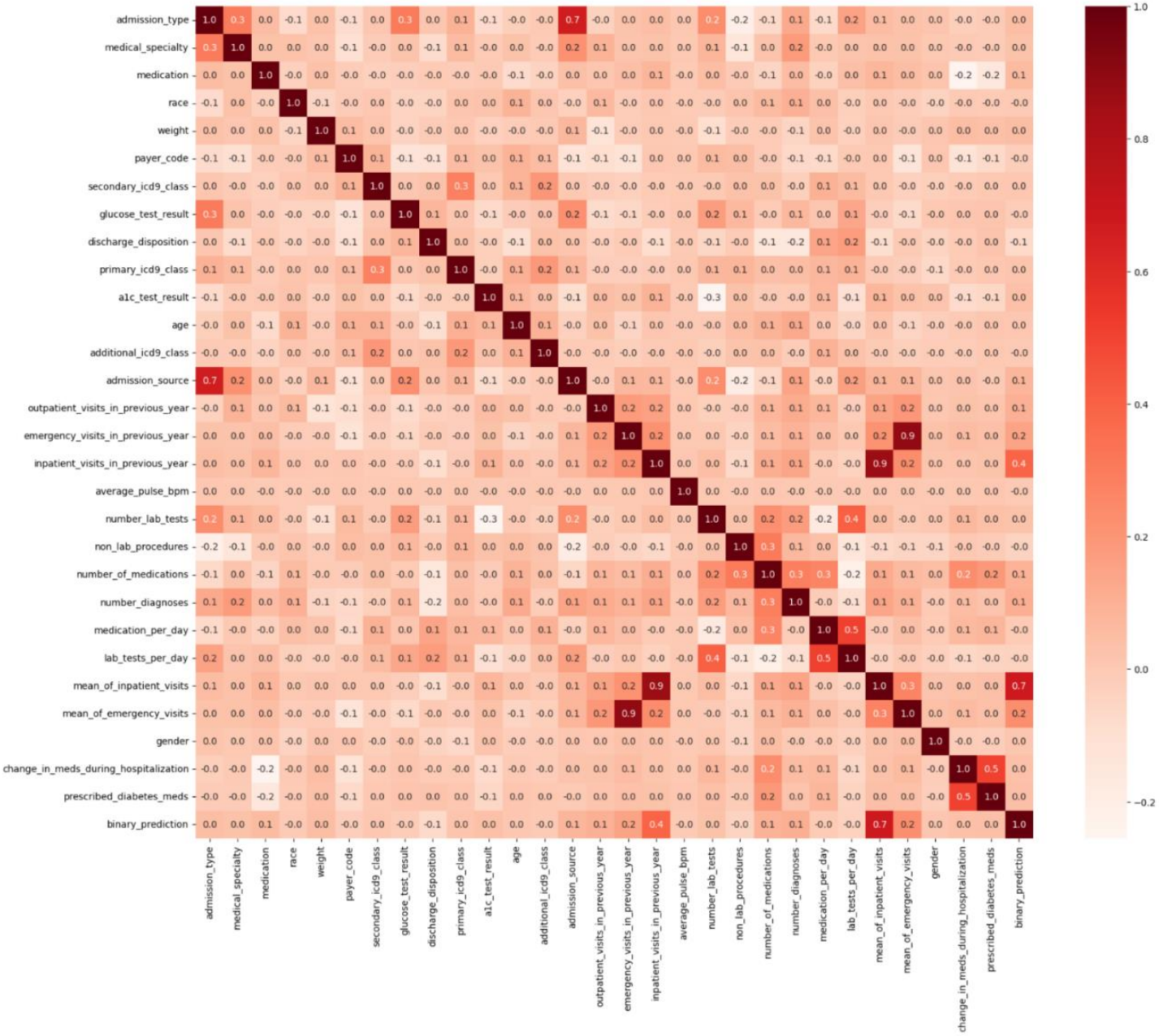

Logistic Regression (L1)
Logistic Regression (L1) (AUC = 0.6780)
Logistic Regression (L2)
Logistic Regression (L2) (AUC = 0.6783)
Logistic Regression (L1 RUS)
Logistic Regression (L1 RUS) (AUC = 0.6756)
Logistic Regression (L2 RUS)
Logistic Regression (L2 RUS) (AUC = 0.6756)
Decision Tree
Decision Tree (AUC = 0.6821)
Decision Tree (RUS)
Decision Tree (RUS) (AUC = 0.6821)
Random Forest
Random Forest (AUC = 0.6969)
Random Forest (RUS)
Random Forest (RUS) (AUC = 0.7019)
HistGradient Boosting
HistGradient Boosting (AUC = 0.6954)
HistGradient Boosting (RUS)
HistGradient Boosting (RUS) (AUC = 0.6996)
SVM
SVM (AUC = 0.6686)
SVM (RUS)
SVM (RUS) (AUC = 0.6682)
SVC
SVC (AUC = 0.6869)
SVC (RUS)
SVC (RUS) (AUC = 0.6872)
BernoulliNB
BernoulliNB (AUC = 0.6723)
BernoulliNB (RUS)
BernoulliNB (RUS) (AUC = 0.6748)
GaussianNB
GaussianNB (AUC = 0.6355)
GaussianNB (RUS)
GaussianNB (RUS) (AUC = 0.6393)
MLP (RUS)
MLP (RUS) (AUC = 0.6837)
KNN (RUS)
KNN (RUS) (AUC = 0.6442)
True Positive Rate
False Positive Rate

[AUC]

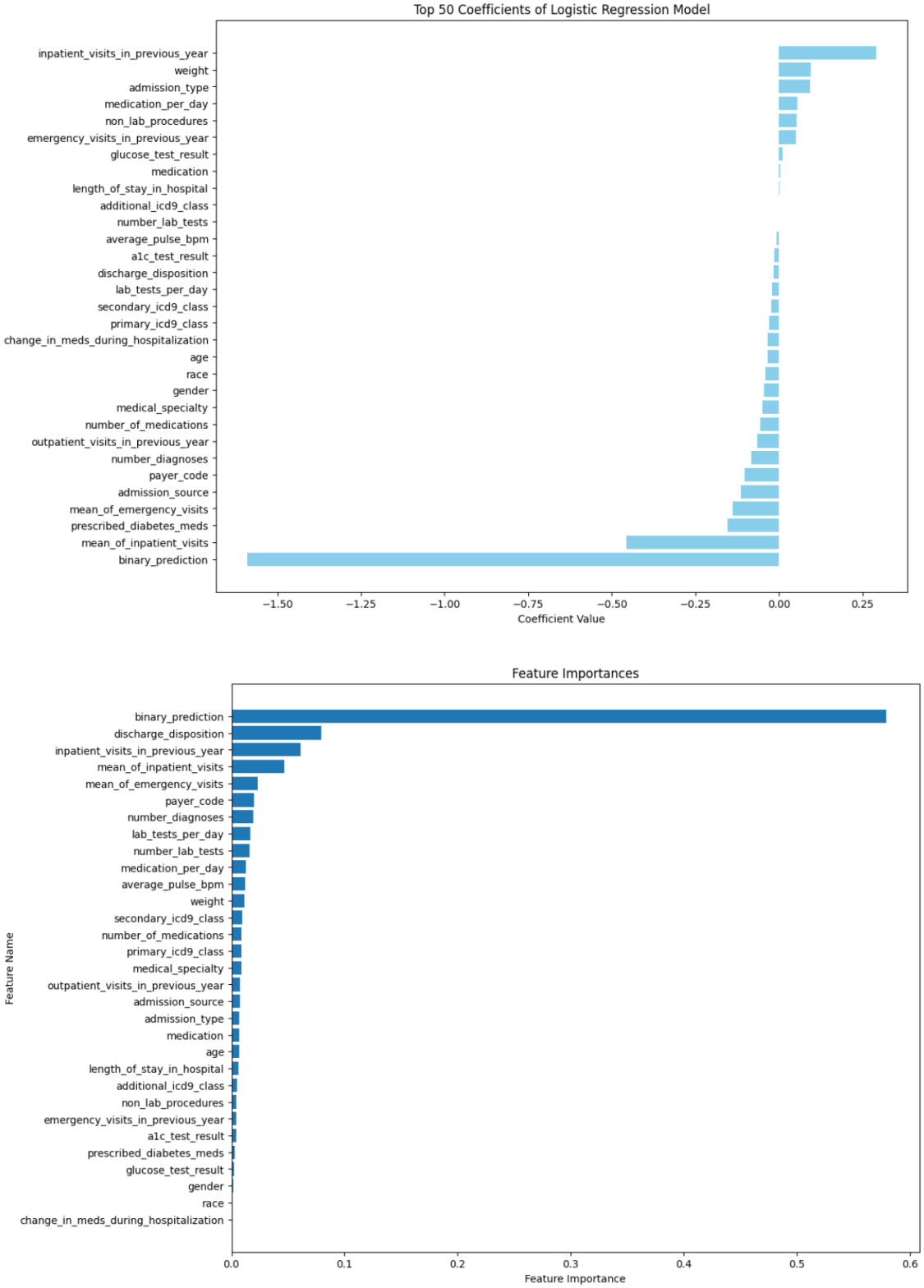
	Readmission 1 True	Readmission 0 True
Readmission 1 Pred	0.065343	0.173077
Readmission 0 Pred	0.046252	0.715328

Confusion Matrix 1

https://github.com/scikit-learn/scikit-learn/issues/26768
KNN Error 1



Correlation Matrix 2



Feature Importance 1

```
# param_grid = {
#     'n_estimators': [70,80,90],
#     'max_depth': [15,17,19],
#     'criterion': ['gini'],
#     'max_features': [30],
#     'min_impurity_decrease' : [0.0001],
#     'min_samples_split' : [20,50],
#     'min_samples_split' : [20,50],
# }
# Best Parameters:  {'criterion': 'gini', 'max_depth': 19, 'max_features': 30, 'min_impurity_decrease': 0.0001, 'min_samples_split': 20, 'n_estimators': 90}
# Best F1:  0.6417834200623391
# Train Error F1 0.7464774854236014
# Test Error F1 0.6245084723024217

# param_grid = {
#     'n_estimators': [90,110,130],
#     'max_depth': [19,20,21],
#     'criterion': ['gini'],
#     'max_features': [30],
#     'min_impurity_decrease' : [0.0001]
#     # 'min_samples_split' : [20,50],
#     # 'min_samples_split' : [20,50],
# }
# Best Parameters:  {'criterion': 'gini', 'max_depth': 21, 'max_features': 30, 'min_impurity_decrease': 0.0001, 'n_estimators': 130}
# Best F1:  0.6458281968467185
# Train Error F1 0.7630912108330582
# Test Error F1 0.6286921176823577

# param_grid = {
#     'n_estimators': [130,150],
#     'max_depth': [22,24,26],
#     'criterion': ['gini'],
#     'max_features': [30],
#     'min_impurity_decrease' : [0.0001]
#     # 'min_samples_split' : [20,50],
#     # 'min_samples_split' : [20,50],
# }
# Best Parameters:  {'criterion': 'gini', 'max_depth': 22, 'max_features': 30, 'min_impurity_decrease': 0.0001, 'n_estimators': 150}
# Best F1:  0.6468824935447032
# Train Error F1 0.7643526699699166
# Test Error F1 0.6298638640547758
```

[param grid rf]

param grid rf 1

```
# svc= SVC(class_weight='balanced', random_state=0, kernel='rbf')
# param_grid = {
#     'gamma': ['auto'],
#     'C' : [0.1,1,2,3]
# }
# Best Parameters:  {'C': 3, 'gamma': 'auto'}
# Best F1:  0.6052086283464793
# Train Error F1 0.6937447321487482
# Test Error F1 0.601337462681217
#overfitting but model wants still higher values of c

# param_grid = {
#     'gamma': ['auto'],
#     'C' : [0.01,5,10]
# }
# Best Parameters:  {'C': 5, 'gamma': 'auto'}
# Best F1:  0.6066036178141353
# Train Error F1 0.7278643607766774
# Test Error F1 0.6031508499301774
#overfitting on train data but still improving the test error
```

[param grid svc]

param grid svc 1

	Readmission No True	Readmission >30 days True	Readmission <30 days True
Readmission No Pred	0.445045	0.129562	0.027442
Readmission >30 days Pred	0.081836	0.157004	0.046954
Readmission <30 days Pred	0.012212	0.062746	0.037198

[confusion matrix 2]