NOVA IMS
Information Management School

HOSPITAL READMISSIONS

Project Report

-2023/2024-

Group 12

Ana Maia r20201562
Diogo Silva r20201643
Matilde Santos r20201596
Marta Flor r20201508
Pedro Catarro 20230463

Abstract: In healthcare, one important factor that can help keep performance stable is the analysis of patient readmissions, understanding if there's a gap in care, or if the financial burden is coherent. To understand this, the following study aims to develop a predictive model based on the data available, to estimate if a patient will be readmitted within 30 days (Binary Classification) and predict the timeframe of a patient's readmission (Multiclass Classification). The dataset in study went through data preprocessing with outliers, missing values, and incoherence treatment, the creation of a few variables in feature engineering, and, finally, several methods of feature selection. After this, several model performances were assessed and compared to reach the best final solution, which turned out to be a model made with Random Forest (for both Binary and Multiclass Classification problems). Furthermore, the evaluation metrics from our Multiclass predictions showed a better overall f1 score, and some limitations were encountered like the significant presence of missing data and class imbalance (further explained during conclusion).

Table of Contents

Table of Visualizations

I. Introduction

Healthcare is a driving factor in society's development, as it ultimately keeps its members able to perform their tasks. One of the factors that can help keep healthcare performing is the analysis of patient readmissions, this can help understand if there's a gap in care or if the financial burden could be lowered.

To understand how hospital readmissions are affecting healthcare, in this project, we aim to develop a predictive model based on the data available to estimate two targets:

- Binary - If a patient will be readmitted to the hospital within 30 days of being discharged.
- Multiclass - Predict the timeframe of a patient's readmission based on 3 classes ("No", "<30 days", ">30 days").

The development of this project consisted of 5 main steps: (1) Data Exploration, (2) Data Pre-Processing, (3) Predictive Modeling, (4) Testing, and (5) Deployment.

Firstly we imported, analyzed, split, and pre-processed the dataset according to what we learned in class but also self-learned content. Then, we applied different predictive models to find the one best suited for our context (the one that gave us the best f1-score). To conclude we finalized our project by testing our now-trained model and found our conclusions. This description applies to both binary and multiclass classifications. Firstly, it was applied to the binary one, and then to multiclass, adapting the code on the way.

A study by the Canadian Medical Association Journal (CMAJ) ([R1]) on the number of hospital readmissions deemed avoidable, found this value to be around 27%, a number which they say is lower than what was expected by previous studies. They justify this due to "(...) the subjectivity of the outcome, as well as differences in study characteristics (...)", meaning it's hard to predict such variables consistently. This study gives us a point of view on the complexity of this subject and some aspects to be aware of.

This report aims to describe all steps in a more detailed way to give a full perspective on the project and present our conclusions and observations.

II. Data Exploration

The first stage of the project was to understand the main objectives and discuss the methodologies and approaches that were going to be used. Subsequently, we started the process of better understanding our data. For this purpose, we analyzed our data set, extracting some information, such as the columns' unique values to check the presence of odd values and, also, the cardinality (In the data pre-processing section, we'll go into more detail about this topic). The dataset’s shape and summary statistics were also carried out to verify inconsistencies.

Later, 'encounter_id' was defined as an index since it is mentioned in the project description that the present features are associated with each encounter. Then, we checked whether our dataset had any duplicates, which we found it didn't.

Furthermore, we separated our metric and non-metric features so we could perform distinct analyses tailored to each set. Accordingly, we analyzed the distribution and boxplots of the numerical variables and the histograms of the categorical variables to get a better understanding. We observed that the majority of the numerical variables had a skewed distribution, indicating that there was a possibility of

outliers. Finally, we checked the percentage of unique values of the target variables:

- In '*readmitted_multiclass'*, we had approximately 54% of "No", 35% of ">30 days", and 11% of "<30 days".
- In *'readmitted_binary'*, we get approximately 89% "No" and 11% "Yes".

This indicates that we have an imbalanced dataset, which is important information to further decide the methodologies applied to the dataset.

Furthermore, we made additional visualizations to explore the relationships between the variables. We could conclude that patients between 50 and 90 are the predominant age group, being [70-80) the group with the highest number of readmissions (Annex Table 1). Additionally, people belonging to the [20,30) have a considerably higher average number of inpatient visits in the previous year. We also could observe a downward trend in the average number of emergency visits after the [20, 30) age group (Annex Table 3).

As for diagnoses, the most common condition among patients in the dataset is diagnosis codes 428, 276, and 401 representing 'Heart Failure', 'Disorders of Fluid, Electrolyte, and Acid-Base Balance', and 'Hypertension', respectively. These conditions appear frequently in all three categories (primary, secondary, and additional diagnoses). Additionally, the Diseases of the Circulatory System are the ones with the most hospital readmissions.

Regarding patients readmitted in less than 30 days, it's more common to have considerably higher inpatient visits in the previous year, i.e., visits with the intention to stay overnight, which could indicate that the variable may be a good predictor (Annex Table 2).

Additionally, as the length of stay increases, there is a corresponding increase in the number of inpatient visits, while the frequency of other types of visits remains relatively stable. As expected, the number of medications administered and lab tests conducted also increases as the patient's length in the hospital increases (Annex Table 5).

III. Data Pre-Processing

a. Incoherences

In medicine, the *primary_diagnosis* refers to the actual condition the individual is being followed healthcare-wise. The *secondary_diagnosis* and *additional_diagnosis* describe any other condition suffered in the same period as the primary. Given this, both cannot take equal values. Therefore, when this situation occurred, the repeated values were replaced by "No Diagnosis".

Discharge_disposition refers to the "destination given to the patient after being discharged" [R2]. While checking the columns' unique values, it was noticed that this column had some value as "Expired" referring to deceased people. Hence, these records were eliminated as this project's goal is to predict if a patient will be readmitted to the hospital, so they are irrelevant to the analysis. In this way, as we considered that it would not be possible for the patients to be readmitted to the hospital when we were predicting the readmissions in the test dataset, we manually assigned the target a value of 0. This way we are enabling the model to focus on what is important, regarding the objective of the project.

b. Train & Validation Dataset Split

Before starting the pre-processing steps, it is essential to divide the data set into training and validation subsets, aiming to obtain a more efficient evaluation of the models' performance and to allow comparisons between them. In our approach, we have implemented a 70/30 split to assign 70% of the data for training and the remaining 30% for validation. Additionally, during the splitting process, we included the stratification parameter (stratify = target) to ensure that the distribution of target classes remains consistent across the training and validation sets. The random_state was set to 42 in order to ensure consistency and reproducibility when running multiple times.

c. Outliers

Outliers are data records that differ greatly from the rest of the data. It is important to analyze them as they can have a large impact on statistical analysis and can lead to less effective models. We have considered 2 approaches to deal with outliers: **(1) Manual Approach**, where we looked at the distributions of numerical variables with detail, through Boxplots (Figures 6 and 7) and Histograms made. Based on them, we did a manual filtering considering the observations that we wanted to keep. For example, with this filter, we would only consider keeping the *number_of_medications* that are below 65, since we consider that it is unlikely that one will be administered with more than this value of different medications, during the encounter. With this method, 0.35% of the data would be removed. **(2) Interquartile Range**

method, where we computed the quartiles (Q1 and Q3) of each variable, defined upper (Q3 + 1.5*IQR) and lower (Q1 - 1.5*IQR) limits and the values that fell above, or below these 2 values, respectively, were considered outliers and would be removed, after the filter was applied. With this method, 33.88% of the data would be removed, which is not a reasonable solution since we want to keep as much data as possible to maintain the integrity of the data.

After this, we did a filter to combine both methods, so that only the records that were considered outliers in both approaches would remain as outliers. After some consideration and analyzing the results of the f1 score, we decided to do variable distribution transformations (further explained in the feature engineering section), instead of eliminating the observations considered outliers.

d. Feature Engineering

Once the data has been treated, the following task is to improve and optimize the quality of our features, attempting to extract relevant information to discover patterns and enhance the performance and interpretability of the models. In this way, we applied various transformations to the variables, namely category aggregation, transformation of categorical variables into dummy/numerical variables, and logarithmic and square-root transformations.

During the data exploration phase, we noted significant categorical variables with numerous categories. As a result, we opted to group these categories to generalize the classes, aiming to make the model less prone to overfitting and to improve its learning capabilities. Accordingly, we applied the following category grouping strategies:

- **Diagnosis Variables (*primary_diagnosis*, *secondary_diagnosis*, *additional_diagnosis*):** These variables were coded as the first three digits of the International Classification of Diseases, Ninth Revision (ICD-9). Hence, we grouped them based on the chapters outlined in the ICD-9-CM

Chapters List [R3]. This transformation reduced the number of categories from approximately 700 to 18, providing a more generalized representation.

- **Payer Code (*payer_code*):** We aggregated categories with less than 5% of observations into an "Other" category, which reduced the number of classes from 18 to just 4 (“No Insurance”; “MC”; “Other”; and “HM”).
- **Medical Speciality (*medical_speciality*):** Categories related to surgery, pediatrics, psychiatry, and women's health were grouped, resulting in a reduction of 68 to 44 categories (Example: The categories 'Surgery-Cardiovascular/Thoracic' and 'Surgery-General' were grouped into 'Surgery').
- **Discharge Disposition (*discharge_disposition*):** We generalized the categories into broader groups, namely, "Within Facility Transfer," "Another Facility Transfer," "Home Discharge," "Still patient or expected to return for outpatient services," and "Other Situations." This transformation reduced the number of categories from 22 to 5.
- **Admission Source (*admission_source*):** The categories were organized into broader groups, particularly “Emergency Room”, “Transfer”, “Referral”, and “Other”, decreasing the number of categories from 15 to 5.
- **Race**: We renamed the categories “Hispanic” and “Asian” to “Other” as they represented less than 2% of observations. This decision resulted in a broader classification containing only “Caucasian”, “AfricanAmerican” and “Other”.

Another transformation step we performed was to convert categorical variables into numerical features primarily to facilitate model interpretability. Accordingly, we transformed into dummy variables all features with only two classes, specifically, ***gender*** which was renamed as “Female”, where 1 indicates the patient is female and 0 otherwise; ***change_in_meds_during_hospitalization***, which was coded as 1 if there was a change and 0 if there was no change; and, ***prescribed_diabetes_meds***, represented as 1 for patients prescribed medication for diabetes and 0 otherwise. Additionally, we converted ***age*** into a numerical format by calculating the average values within each class (e.g., the age range '[10,20[' was transformed into 15).

Alongside applying transformations to the existing features, we have considered creating new variables aiming to capture relevant aspects of the data that may have predictive value on hospital readmissions. These variables were: ***total_visits_in_previour_year*** represents the total number of times a patient visited the hospital in the preceding year, providing a valuable metric for understanding the individual's healthcare utilization in the previous year. To derive this metric, we aggregated the counts of inpatient visits (*inpatient_visits_in_previous_year*), outpatient visits (*outpatient_visits_in_previous_year*), and emergency visits (*emergency_visits_in_previous_year*).

Moreover, considering that hospitalized patients with diabetes may be at higher risk of readmission compared to those without diabetes [R4], we attempted to distinguish between these two groups. Accordingly, we created two variables: ***type2*** and ***diagnosed_diabetes***. The first variable (***type2***) is a dummy variable, assuming a value of 1 if the patient has type 2 diabetes and 0 otherwise. We determined this by examining each patient's prescribed medications and identifying whether they had the presence of Metformin (a drug that has been used for many decades to treat type 2 diabetes [R5]). As for the ***diagnosed_diabetes***, it also operates as a dummy variable, indicating whether the patient was diagnosed with diabetes during that encounter. To create this feature, we referred to the IC9 codes and verified

whether the patient was diagnosed with 250 or 250.XX (e.g. 250.01, 250.02, …) [R6], which represents a diabetes diagnosis.

Additionally, regarding the significance of emergent/urgent admissions and the history of recent hospitalization as high-risk factors for hospital readmission [R7], we created two variables that attempt to reflect these issues: ***admitted_urgent***, a dummy variable that indicates whether the patient was admitted to the hospital as an Emergency or Urgent admission; and, ***total_encounters***, that indicates the number of times the patients were admitted to the hospital.

As mentioned previously, we decided to apply logarithmic and square-root transformations to the variables that presented a skewed distribution in order to reduce the impact of the outliers' values. Hence, we apply the logarithm to *length_of_stay_in_hospital* and *number_diagnoses***.** For the variables that took a value of 0, it wasn't feasible to perform the logarithm, so we employed the square root transformation to *number_of_medications*, *outpatient_visits_in_previous_year*, *inpatient_visits_in_previous_year*, *emergency_visits_in_previous_year*, and *non_lab_procedures*.

e. Missing Values

After a deeper understanding of our dataset and after identifying the unique values of our data, we noticed some unusual values (“?”, “Unknown/Invalid”, “Not available”, “Not Mapped”, “ “ and “[]”) in variables like *payer_code*, *admission_type*, *medical_specialty*, and others. For simplification purposes, we decided to replace all these values with *nan* type, with the objective of treating each of these variables logically, considering the context of our dataset. After the replacement, the variables *race*, *gender*, *age*, *weight*, *payer_code*, *admission_type*, *medical_specialty*, *discharge_disposition*, *admission_source*, *primary_diagnosis*, *secondary_diagnosis*, *addition_diagnosis*, *glucose_test_result*, *a1c_test_result* and *medication* presented missing values. We started to treat missing values in the following variables:

- ***payer_code***: We attributed the value “No Insurance”, as we considered that could be a possibility of the patient not having insurance.
- ***glucose_test_result*** and ***a1c_test_result***: We attributed the value “Not Taken” since the project description mentioned that there were patients who did not undergo these tests.
- ***admission_source***: We attributed the value “Not Mapped” because we believe that not all options of admission sources are present in these feature values.
- ***medication***: We considered the missing values as “No Medication” since the patient may not have been prescribed any medication.
- ***secondary_diagnosis*** and ***additional_diagnosis***: We attributed the value “No Diagnosis” as the patient could only have one diagnosis.

After this treatment, the variables, *race*, *gender*, *age*, *weight*, *admission_type*, *medical_specialty*, *discharge_disposition,* and *primary_diagnosis* presented missing values. Regarding the variable *weight*, the percentage of missing data exceeded 50%, so we decided to remove it from our dataset since it would not add any valuable information and its imputation would be biased as we only had 3% of data available. Additionally, *medical_specialty* also had a high percentage of missing values (47%) but was still relatively below the threshold considered (50%). Despite this, the variable was kept, as in predicting hospital readmissions context, specifying the medical specialty on which the patient was admitted could contain significant information related to the patient readmission, which we will later evaluate in feature selection to achieve a higher grade of certainty. For the rest of these variables, we tried 2 missing value

treatment approaches: **(1) KNN imputation** (to apply this technique, we had to scale our dataset previously using Target Encoding and MinMax Scaler, while preserving the missing values), **(2)** imputing the **mode** of each variable. As the final solution, we decided to apply the KNN imputation since it gave us relatively better results when predicting our target (Annex Tables 3 and 4).

f. Data Scaling

Considering most machine learning algorithms that we will perform, use mathematical operations that require numerical data, it is needed to transform the categorical variables into numerical ones. Some options were considered, namely, One-Hot Encoding, Binary Encoding, and Target Encoding, but only the last two were used. The method of Binary Encoding was used to convert binary variables into dummies (0 or 1), as was already explained in the feature engineering section. Target Encoding was used on the remaining categorical variables based on the presence of high-cardinality features in the dataset. This was also what prevented One-Hot Encoding from being used. Also, according to the literature, "Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features" [R8].

Following Target Encoding, it was necessary to apply scaling to the numerical and categorical variables so that the features contribute equally to the models, preventing variables with larger values from having a greater influence. In this way, it was necessary to apply MinMax scaling to preserve the data distribution. This converts the values encoded previously into a range between 0 and 1.

g. Imbalance Dataset Approach

Another important thing to determine was if the dataset was balanced. As mentioned in Data Exploration, both targets' classes are not balanced. To solve this problem several methods can be applied, namely, **Over-Sampling** (such as SMOTE), **Hybrid approach** (SMOTEENN, and SMOTE Tomek), or **Under-Sampling** (such as Random Under-Sampling). In the end, SMOTE Tomek was chosen because according to the literature, to solve the problem of imbalanced medical data "The results show that the method of SMOTE combined with Tomek links technique is much superior compared with that of using only SMOTE." [R9] and " (…) using Smote-Tomeklink can improve the performance of the random forest method (…)" [R10].

We also considered using the parameter "class_weight = balanced" to handle this issue, whenever available in the model parameters. When in use, the dataset was not balanced using any technique previously mentioned. Later in the report, particularly in the Model Assessment section, we compare the performance of both methods.

h. Feature Selection

Feature Selection is the process of identifying pertinent characteristics and eliminating redundant, noisy, or unnecessary data. It consists of applying statistical models that identify the features most important for us to keep, which we decide based on the results of several methods. Through this technique, predictive algorithms are sped up, and their performance and comprehensibility are increased, which improves results and leads to a better understanding of the target subject. Therefore, in this phase of the project, our goal is to select only the relevant features to help the models perform better.

Firstly for the **Binary Classification**, we started by checking the variable's **variance** and assumed for elimination of the ones whose coefficient was below the threshold (0.001). After this we applied Filter

Methods, meaning that they are independent of any learning method, and they rank features based on a certain evaluation criterion. The first Filter Method used was **Spearman's rank correlation coefficient**, this method draws a graph that shows the correlation between all independent variables and between those and the target variable, Figures 8 and 9 illustrate the results obtained. In this way, we considered the elimination of variables that presented a correlation value with the target below |0.05|, as this can suggest that the variable has low predictive power, and above |0.8| for correlation between the independent features because it can suggest that there is potential redundant information. The second and final Filter Method applied was the **Chi-Square Test**. This method was applied to non-metric features using a 5% significance level.

Even though the results from these methods are important to identify relevant features for the model, they do not consider interaction outcomes between different variables nor with the classifier, hence the implementation of a well-known Wrapper-type algorithm, the **Recursive Feature Elimination (RFE)**. In this method, a different machine learning algorithm (Logistic Regression in our case) is used at the core of the process. This is the piece of code that needs the biggest computational power in our notebooks, becoming its disadvantage, another downside is its susceptibility to over-fitting. After analyzing the behavior of the f1 score for different numbers of features (depicted in Figure 12), we established retaining 12 features since we wanted the least overfitting value combined with the highest f1 score while prioritizing the smaller number of features to avoid having sparse data (curse of dimensionality).

For the **Embedded Methods**, we applied the **Lasso Regression**, which performs feature selection intrinsically. This method is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the resulting statistical model. Figure 14 in the Annex depicts the feature importance output of this method. Features that presented a coefficient between |0.15| were assumed for elimination. The second Embedded-type method we applied was the **Decision Tree**. During tree construction, it only divides on features that generate the largest gain in node purity, meaning the worst features aren't even iterated. The threshold used for the feature importance was 0.015. The outcomes of this selection process are illustrated in Figure 16, located in the Annex.

After doing this for Binary Classification, we adapted the methods and logic to the **Multiclass Classifier**. Firstly, we changed the variance threshold to 0.1. Following this, during the implementation of **RFE**, the number of features was set to 11, after analyzing Figure 13. For the **Decision Tree** algorithm (Figure 17), features with importance values below 0.025 were assumed for elimination. Besides these changes, the remaining methods (Spearman Correlation and Lasso Regression) kept the same thresholds, and their results are depicted in Figures 10, 11, and 15 respectively). We considered these thresholds because we wanted to be as critical as possible in order to retain the truly important features regarding our objective. In this way, the model will be more prone to perform better in unseen data.

To conclude the Feature Selection phase of the project, we developed an algorithm that searched all features previously assumed for elimination and removed them from the dataset in case they were present in the results of half (3) or more of the methods used. In Figures 18 and 19, two tables illustrating the combination of all methods, and the final selected features for each classification are the following:

- **Binary:** *'age', 'emergency_visits_in_previous_year', 'inpatient_visits_in_previous_year', 'medical_specialty', 'discharge_disposition', 'length_of_stay_in_hospital', 'number_of_medications', 'number_diagnoses', 'medication', 'total_visits_in_previous_year',*

*'primary_diagnoses_categories', 'secondary_diagnoses_categories', 'additional_diagnoses_categories', 'total_encounters'**.*

- **Multiclass:** *'age', 'outpatient_visits_in_previous_year', 'emergency_visits_in_previous_year', 'inpatient_visits_in_previous_year', 'medical_specialty', 'discharge_disposition', 'length_of_stay_in_hospital', ''number_lab_tests', 'number_of_medications', 'number_diagnoses', 'medication', 'total_visits_in_previous_year', 'primary_diagnoses_categories', 'secondary_diagnoses_categories', 'additional_diagnoses_categories', 'encounter_count'**.*

*These variables have different names however they have the same meaning.

IV. Modeling

Moving on to the modeling phase, we had two different prediction problems: predicting binary and multiclass hospital readmissions. Accordingly, we conducted a model exploration employing diverse algorithms to address these issues.

Given that algorithms perform differently based on the dataset's characteristics, we have considered a set of seven models, each adopting different prediction approaches. These models encompassed a variety of algorithms, including **Logistic Regression**, **Naive Bayes**, **K-Nearest Neighbors (KNN)**, **Multi-Layer Perceptron (MLP)**, **Decision Trees**, **Random Forest**, **Support Vector Classification (SVC)**, and **Gradient Boosting**.

Upon selecting the models, the following step was to **optimize the parameters** of each model, aiming to maximize the validation dataset performance while minimizing overfitting, i.e., ensure that our models effectively learn the underlying patterns in the data and can apply them to unseen datasets. Hence, we used the **Random Search method** to optimize the parameters, which consists of randomly selecting parameters within a predefined Search Space and returning the ones that obtain the best results. Regarding the Search Space, we determined the parameters deemed necessary and most appropriate for our problem. For this phase, we also considered the Grid Search method, which exhaustively evaluates all possible combinations within a Search Space parameters. However, considering the parameters considered and the size of our data set, it would become computationally expensive and time-consuming, leading us to choose the Random Search method for parameter optimization. The Annex Tables 1 and 2 display an overview of the parameters we tested and the best solution obtained by the Random Search.

Having the parameters established, as mentioned above, we tried two different approaches to deal with the class imbalance: **class_weights** and **SMOTE Tomek**. As a result, we evaluated the performance of each model under each of these scenarios. However, it is important to acknowledge the absence of the class_weights parameter in some models. Hence, we could only use this technique for the models that included the parameter, i.e., Logistic Regression, Decision Trees, Random Forest, and SVC.

Finally, in an attempt to improve the overall results, we applied two **ensemble techniques**: **Gradient Boosting**, in which the model is enhanced by gradually adding more models to the ensemble [R11], and **Random Forest**, which combines the output of multiple decision trees to reach a single result.

a. Binary Classification Assessment

When evaluating models for predicting binary targets, we prioritize **F1-Score** as our primary metric, as it provides a harmonious balance between precision and recall. Furthermore, given the unbalanced nature of our dataset, accuracy alone can be misleading, as it can favor models that simply predict the majority class, while F1-Score can guarantee that our model not only captures the majority class accurately but also

can predict the minority class. Additionally, we used a **stratified cross-validation approach with 5-fold validation**, meaning the dataset is subjected to training and validation five times, providing an equal representation of the target in both sets. Then, the final F1-Score is calculated as the average of these iterations, providing a robust assessment of the model's performance.

As mentioned above, we attempted to train each model with the dataset generated with SMOTE Tomek and the imbalanced dataset but applying class_weights = 'balanced'. As can be seen in Annex Table 5, the values we obtained using the dataset with SMOTE Tomek reveal a significant presence of overfitting in general. As this technique creates new observations in the minority class based on the data, this may have introduced noise into the dataset, causing the model to not learn as efficiently. For example, in the MLP model, we obtained an F1-score in the minority class (1) of 0.22 in the validation set and 0.84 in the train set contrasting with the values of the majority class (0) that has an F1-score of 0.91 and 0.85, respectively, reinforcing our previous statement. On the contrary, when we used the dataset imbalanced with class_weights = 'balanced', the overfitting is significantly lower, as this parameter causes the model to change the errors penalization of each class, assigning a higher weight to the minority class and a lower weight to the majority class, making it pay more attention to the minority class during training. That said, we focused on the top 3 models with the best f1-score when the class_weights method was applied (Table 1). These were **Random Forest**, **Support Vector Classifier**, and **Logistic Regression**.

Models	Cross Validation F1-Score
Random Forest	f1_train: 0.33 | f1_val: 0.3
Logistic Regression	f1_train: 0.31 | f1_val: 0.28
SVC	f1_train: 0.30 | f1_val: 0.28

Table 1: Best 3 Models in Binary Classification

Among the three models, the **Random Forest** model was the best performer with the highest F1 score (Table 1). To complement the **F1-Score**, we used **Kaggle** as a model selection method, providing information on the performance of each model on unseen data. Again, the Random Forest model consistently showed to have the best performance. In conclusion, based on the F1 score, Kaggle Performance, and non-occurrence of overfitting, the model Random Forest was our selected choice.

b. Multiclass Classification Assessment

Similarly to binary classification, we prioritized the F1-score as our primary metric, because of the reasons mentioned above, but since this metric in this type of problem is computed for each class, an average of these individual values has to be made. Since our dataset is imbalanced, the average we chose to evaluate our models was the weighted one, which will consider the number of instances of each class to do the computation. Additionally, we used the **Stratified Cross-Validation Approach with 5-fold Validation** to have a more robust assessment.

We focused on the top 3 models with the best F1 score when the class_weights method was applied (Table 2) since we reached the same conclusion as in the Binary Assessment. These were **Multilayer Perceptrons**, **Random Forest**, and **Decision Trees**. Among these three models, the Random Forest and the MLP model are the best ones among these 3 presented models (Table 2). Since the overall Weighted F1-score presented similar values, we analyzed both confusion matrices with more detail. When looking at the F1-score for each class (Annex Table 6), we could observe that the Random Forest Model has a more consistent performance among the 3 classes, showing significantly less overfitting in the minority class

(Class 0: '<30 days') when compared with the MLP model. Accordingly, we chose the **Random Forest** as the final solution.

Models	Cross Validation F1-Score
MLP	f1_train: 0.59 | f1_val: 0.55
Random Forest	f1_train: 0.58 | f1_val: 0.55
Decision Trees	f1_train: 0.57 | f1_val: 0.54

Table 2: Best 3 Models in Multiclass Classification

V. Conclusions

Hospital readmissions are an important and complex subject to understand since this phenomenon can cause a huge financial burden for public and private payers [R12]. During the development of this project we consulted not only the information provided in classes but we did some investigation as well to better understand the phenomenon under study.

Comparing the Binary and the Multiclass performance metrics, we achieved overall best performance in the Multiclass case. We believe that this might be happening due to the importance of the class separation and information gained. In the binary case we are only predicting if the patient will be readmitted or not within 30 days, and in the Multiclass model, we are also taking into account the possibility of readmission after 30 days. Thus, the model might be benefiting from this additional information enabling it to learn and perform better.

We faced some **limitations** during the course of the project. Although we have put effort in doing the most logical and best-performing missing value treatment, the fact that we had a significant presence of missing data in some variables can still lead to noisy information when doing the respective imputation. Additionally, regarding the class imbalance present in our datasets, there are multiple studies that indicate the negative impact of it in the performance of machine learning models "Imbalance data sets degrades the performance of" (…) "machine learning techniques as the overall accuracy and decision making be biased to the majority class" [R15].

Finally, we encountered a limitation during the feature engineering stage when we desired to create a variable that recorded the number of previous encounters before a given appointment. However, this wasn't feasible as the dataset had no temporal order. To address this challenge, we recommend the implementation (as **future work**) of a more robust database that facilitates the temporal ordering of data, which would contribute to a more comprehensive and insightful analysis.

VI. References

[R1] van Walraven, C., Bennett, C., Jennings, A., Austin, P. C., & Forster, A. J. (2011). Proportion of hospital readmissions deemed avoidable: a systematic review. *Canadian Medical Association Journal*, *183*(7), E391–E402. https://doi.org/10.1503/cmaj.101860

[R2] NOVA IMS Moodle: Log in to the site. (n.d.). https://elearning.novaims.unl.pt/pluginfile.php/210856/mod_resource/content/0/Machine%20Learning%20Project.pdf

[R3] ICD-9-CM Chapters List. (2016). Icd.codes. https://icd.codes/icd9cm

[R4] Rubin, D. J. (2015). Hospital Readmission of Patients with Diabetes. Current Diabetes Reports, 15(4). https://doi.org/10.1007/s11892-015-0584-7

[R5] Samar Hafida, MD. (2020, November 5). Type 2 diabetes: Which medication is best for me? - Harvard Health. Harvard Health; Harvard Health. https://www.health.harvard.edu/blog/type-2-diabetes-which-medication-is-best-for-me-2020110521256

[R6] Aitalohi Amaize, & Mistry, K. B. (2016, April). Table 5, ICD-9-CM diagnosis codes defining diabetes. Nih.gov; Agency for Healthcare Research and Quality (US). https://www.ncbi.nlm.nih.gov/books/NBK368403/table/sb203.t5/

[R7] Rubin, D. J. (2015). Hospital Readmission of Patients with Diabetes. Current Diabetes Reports, 15(4). https://doi.org/10.1007/s11892-015-0584-7

[R8] Pargent, F., Pfisterer, F., Thomas, J., & Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Computational Statistics, 37(5), 2671–2692. https://doi.org/10.1007/s00180-022-01207-6

[R9] Effective prediction of three common diseases by combining SMOTE with Tomek links technique for imbalanced medical data. (2016, May 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/7563084

[R10] Hairani, H., Anggrawan, A., & Priyanto, D. (2023). Improvement performance of the Random Forest Method on unbalanced diabetes data classification using Smote-Tomek link. JOIV : International Journal on Informatics Visualization, 7(1), 258. https://doi.org/10.30630/joiv.7.1.1069

[R11] Alexey Natekin, & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in Neurorobotics, 7. https://doi.org/10.3389/fnbot.2013.00021

[R12] 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). https://doi.org/10.1186/s12874-021-01284-z

[R13] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*(16), 321–357. https://doi.org/10.1613/jair.953

[R14] Bergstra, J., Ca, J., & Ca, Y. (2012). Random Search for Hyper-Parameter Optimization Yoshua Bengio. *Journal of Machine Learning Research*, *13*, 281–305. https://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

[R15] Abd Elrahman, S. M., & Abraham, A. (2013). A review of class imbalance problem. Journal of Network and Innovative Computing, 1(2013), 332-340. https://ias04.softcomputing.net/jnic2.pdf

VII. Bibliography

Kumar, V. (2014). Feature Selection: A Literature Review. *The Smart Computing Review*, *4*(3). https://faculty.cc.gatech.edu/~hic/CS7616/Papers/Kumar-Minz-2014.pdf

Kumar, D. (2020, September 4). *What is LASSO Regression Definition, Examples and Techniques*. GreatLearning. https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/

van Walraven, C., Bennett, C., Jennings, A., Austin, P. C., & Forster, A. J. (2011). Proportion of hospital readmissions deemed avoidable: a systematic review. *Canadian Medical Association Journal*, *183*(7), E391–E402. https://doi.org/10.1503/cmaj.101860

sklearn.preprocessing.TargetEncoder. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.TargetEncoder.html

Sukurat, S. (2023, September 30). Addressing the class imbalance in health data using the resampling technique in Python. https://www.linkedin.com/pulse/addressing-class-imbalance-health-data-using-python-salam-sukurat-/

Marcinrutecki. (2022b, December 29). SMOTE and Tomek Links for imbalanced data. Kaggle. https://www.kaggle.com/code/marcinrutecki/smote-and-tomek-links-for-imbalanced-data

C, B. P. (2022, July 11). How to Detect Outliers in Machine Learning – 4 methods for Outlier Detection. freeCodeCamp.org. https://www.freecodecamp.org/news/how-to-detect-outliers-in-machine-learning/

Da Cunha Abreu, P. M. H. (2022, October 17). The joint-effect of imbalanced and missing data: a challenging task in data analysis. Estudo Geral. https://estudogeral.uc.pt/handle/10316/103003

Sklearn.impute.KNNImputer. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

VIII. Annex

a. Theoretical Explanation of Methods Used

Target Encoding

Most machine learning algorithms use mathematical operations that require numerical data. Therefore, in the presence of categorical data, some methods can be applied to overcome this issue by converting it into numerical features. Target encoding is one example. This method works by grouping the data by each category and counting the number of occurrences of each target. Then, it does the target value average for each category in the column and replaces it with the encoded value.

K-NN imputer

Knn imputer is a method to fill in missing data using the well-known k-Nearest Neighbors classifier. This method computes the k nearest neighbor instances from the one with the missing value, using the distance metric chosen (the default one is nan_euclidean). Then, the missing value will be filled with the mean or weighted mean of the corresponding values in the neighbors' instances, for numerical variables, or the mode, for categorical variables.

SMOTE Tomek

Oversampling is a popular approach to address the problem of imbalanced data. It consists of adding observations to the minority class to balance the dataset with the majority class. The Synthetic Minority Oversampling Technique (SMOTE) is one of the most known oversampling methods. This method consists of adding synthetic instances in the minority class by performing interpolation between the chosen observation and the k-nearest neighbors from the minority class [R13]. Opposite to oversampling, undersampling removes occurrences from the majority class. Tomek Links is an undersampling technique that forms pairs of occurrences from both the majority class and the minority class that are close to each other. Hence, as a result, Smote Tomek is a hybrid method that uses both oversampling and undersampling to solve the imbalance problem. It generates new instances in the minority class using SMOTE and undersamples the majority class by applying Tomek Links, consequently, it can lead to an achievement of better model performance.

SMOTE ENN

Similar to SMOTE Tomek, SMOTE ENN, is also an hybrid method, using SMOTE as the oversampling method. The difference is that it uses Edited Nearest Neighbors (ENN) as an undersampling method. This works by, for each point in the dataset it finds its closest neighbors (using k-nearest neighbor) and if the majority belong to a class different from the one the point in analysis belongs to, they all are eliminated. The goal is to reduce classification mistakes by cleaning the data.

Random Search

In the project, Random Search was used as a hyperparameter tuning approach. Generally, this approach involves establishing a search space as a bounded domain of parameter values and choosing random sample points within that domain. Our decision is supported by a study [R14], which states that randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid. By comparing random search with grid search and manual search, the study proves that random search can identify models with results as good or greater than grid search in a significantly shorter amount of time.

b. Data Exploration

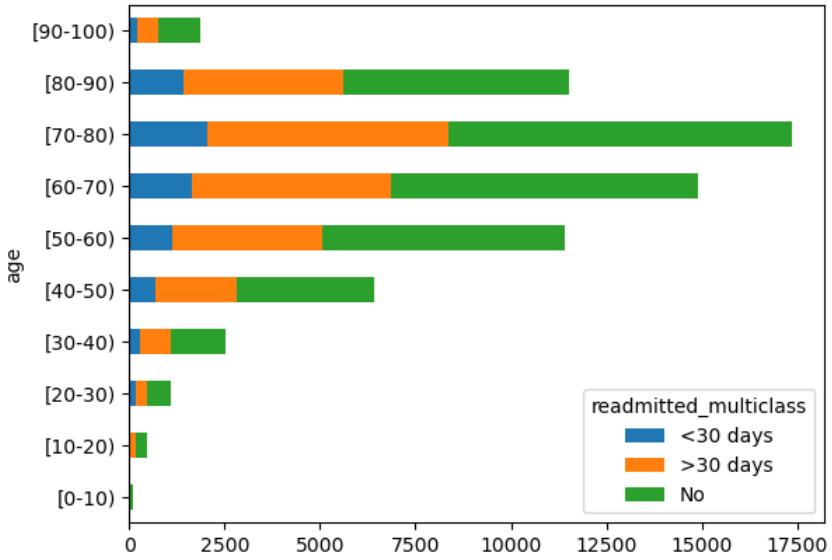

Figure 1: Age/Target Relationship

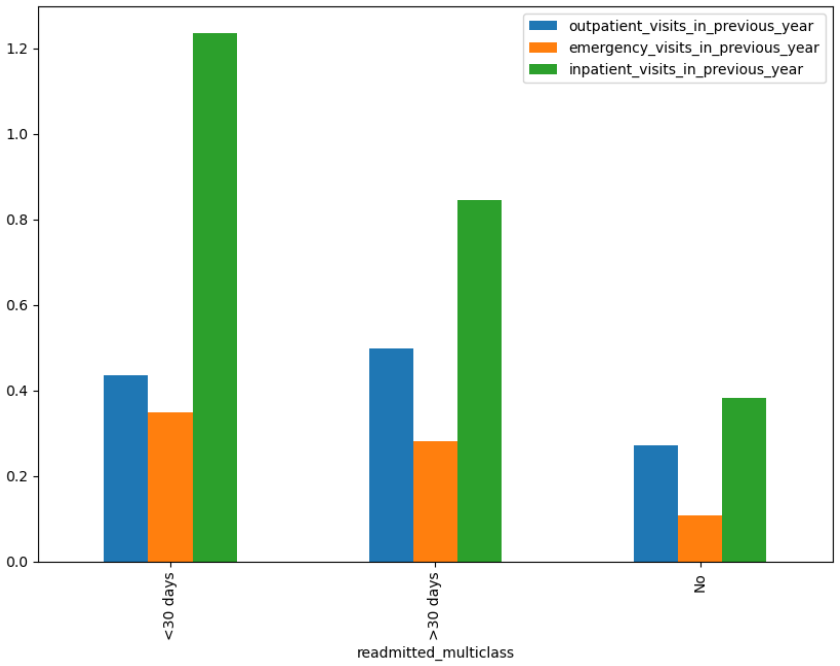

Figure 2: Visits/Target Relationship

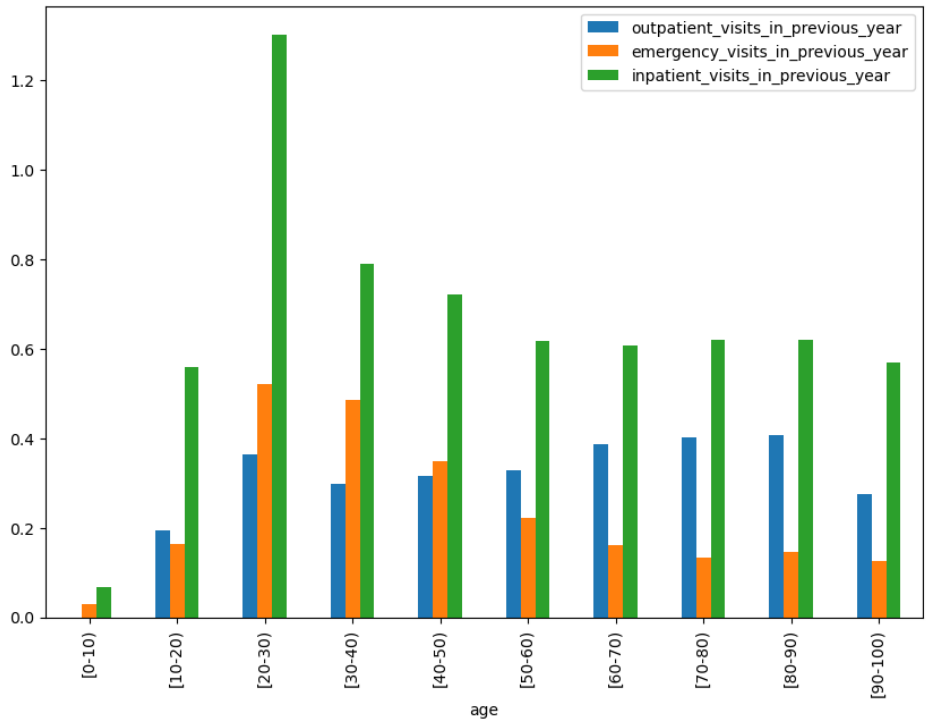

Figure 3: Number of visits per age group

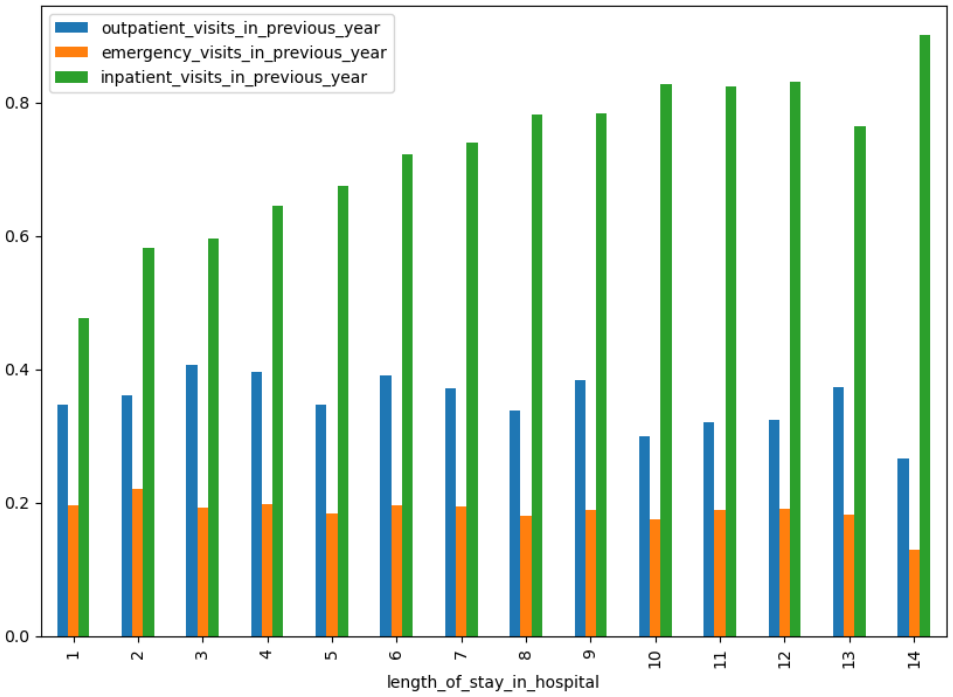

Figure 4: Number of visits per duration of stay

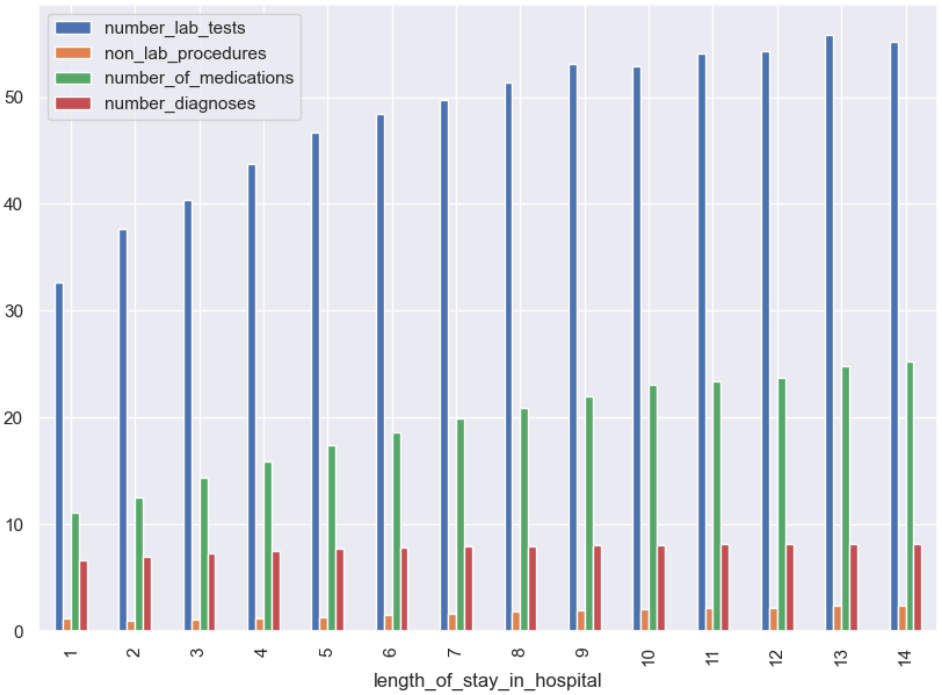

Figure 5: Patient care metrics per duration of stay

c. Outliers

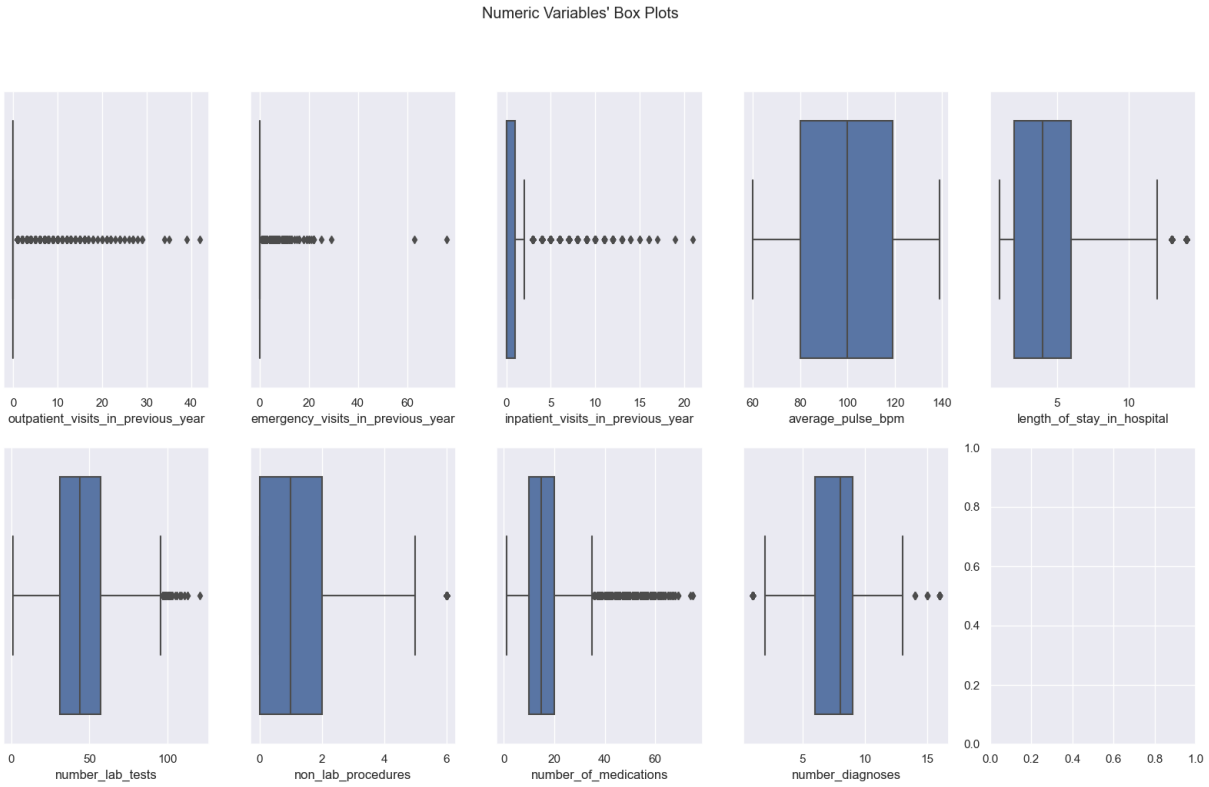

Figure 6: Outliers Box Plot (Binary Classification)

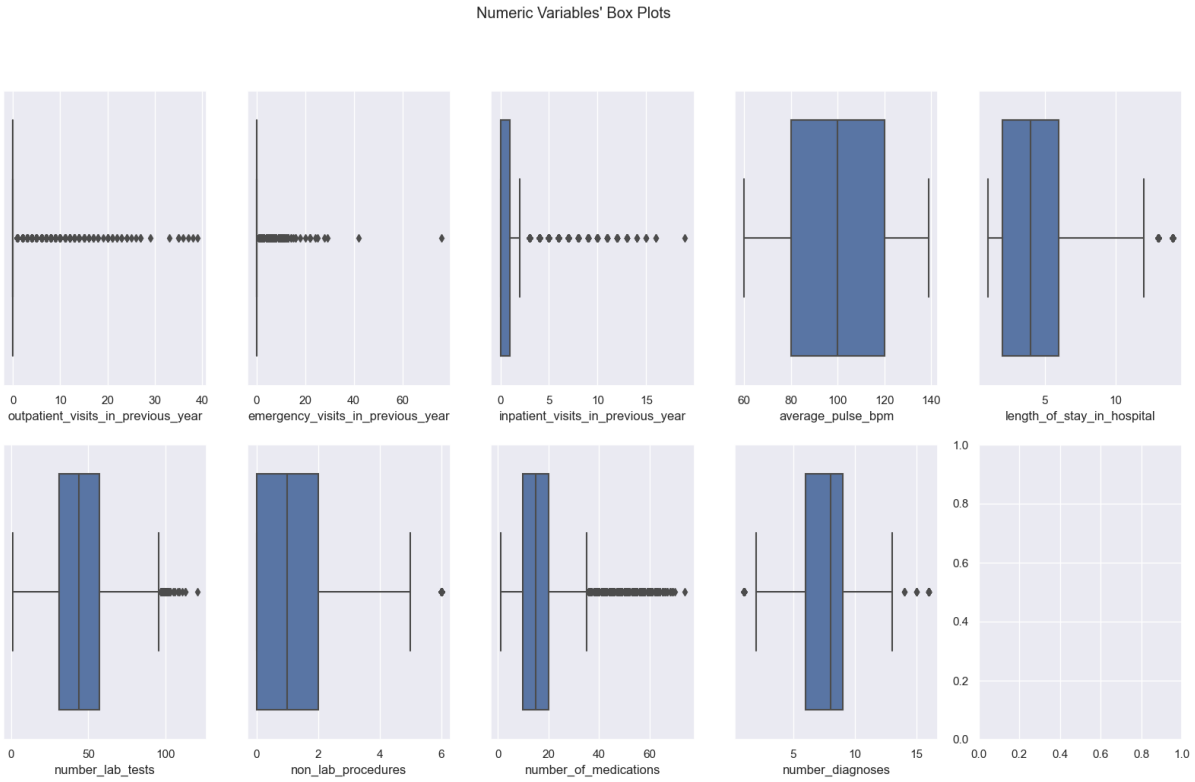

Figure 7: Outliers Box Plot (Multiclass Classification)

d. Feature Selection

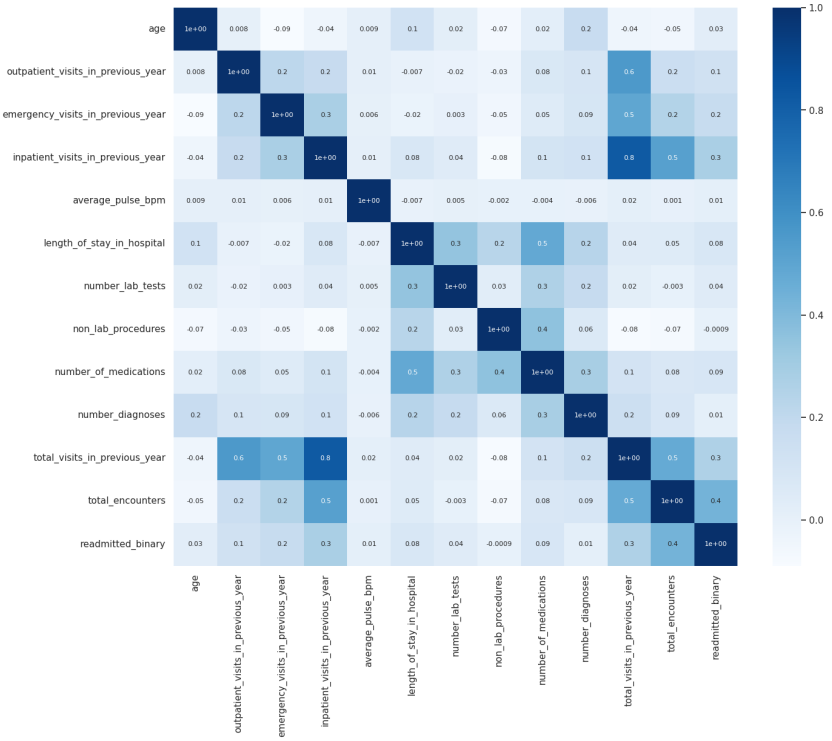

Figure 8: Spearman Correlation matrix metric features (Binary Classification)

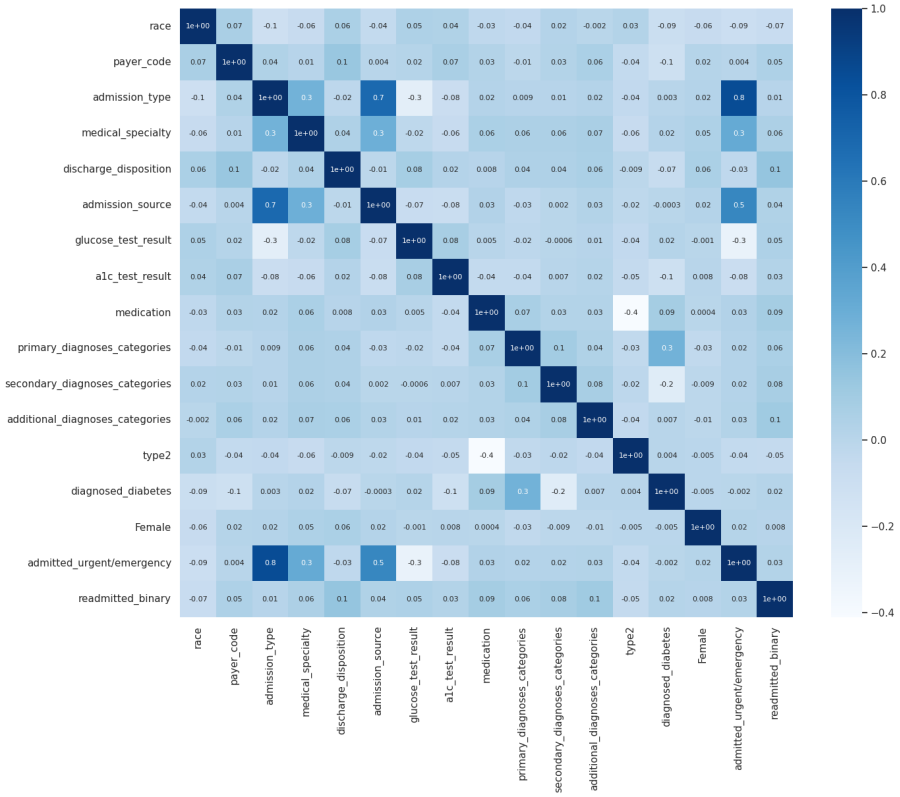

Figure 9: Spearman Correlation matrix non-metric features (Binary Classification)

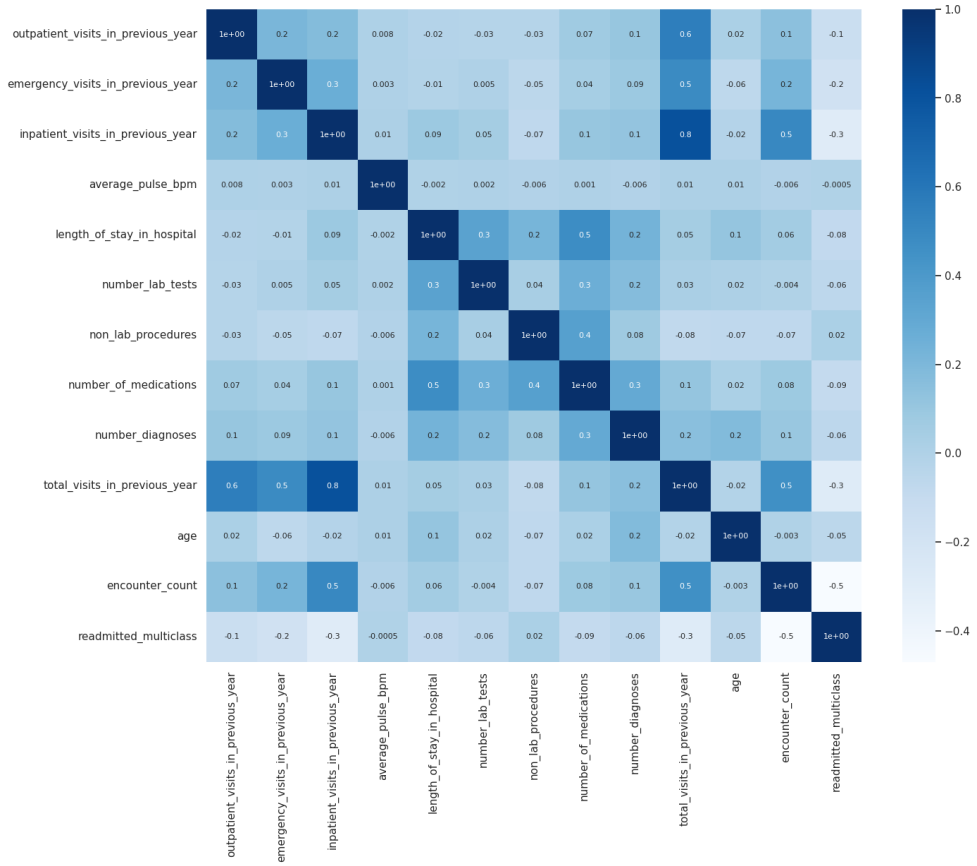

Figure 10: Correlation matrix metric features (Multiclass Classification)

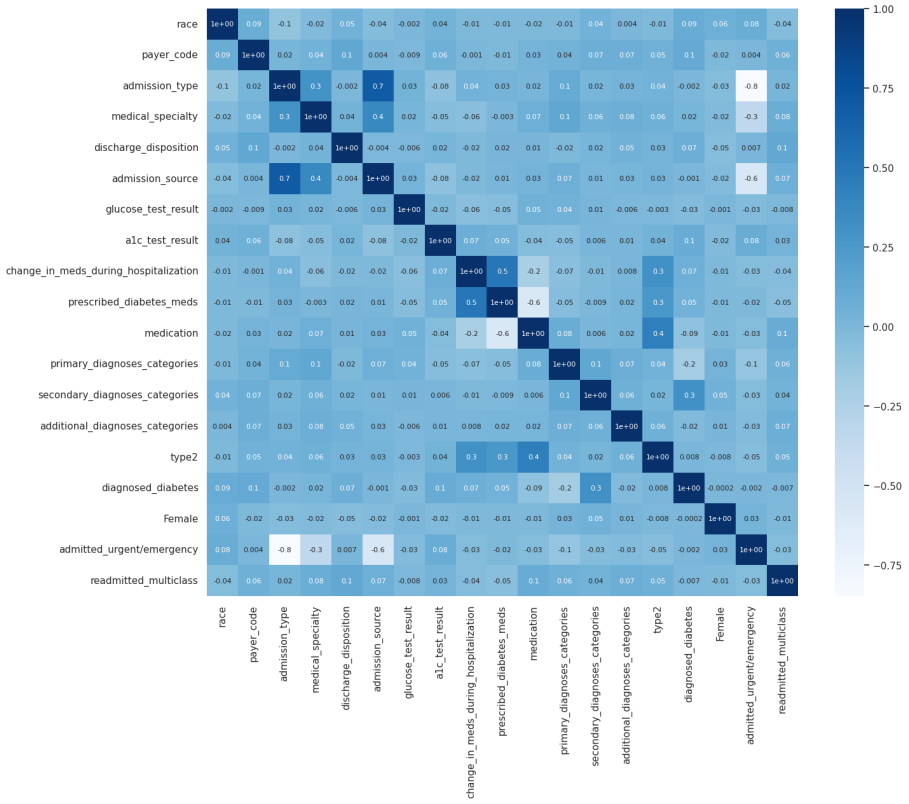

Figure 11: Correlation matrix non-metric features (Multiclass Classification)

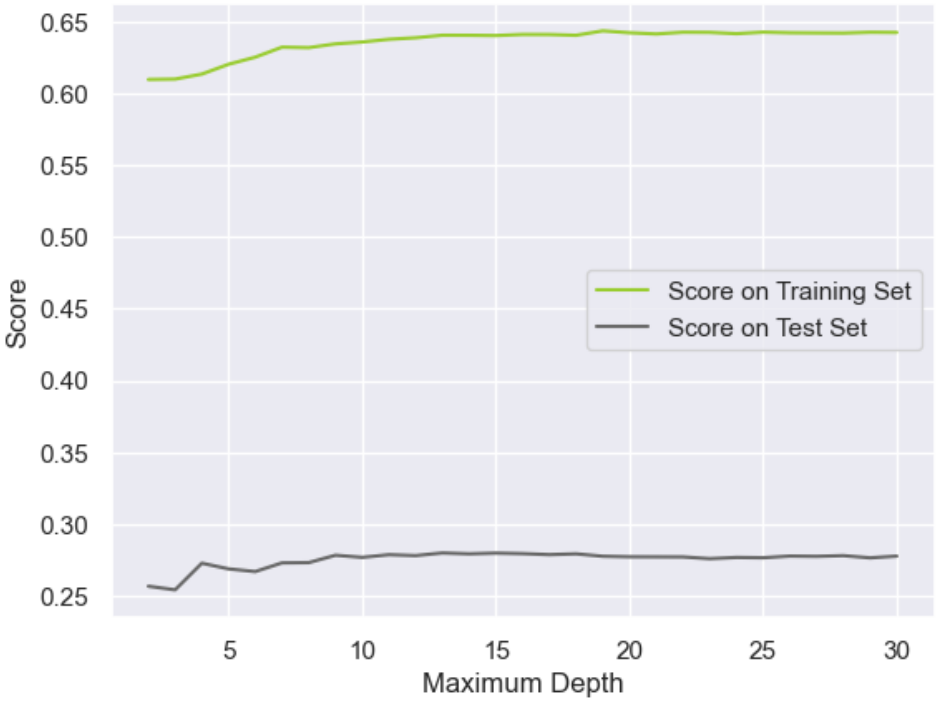

Figure 12: RFE optimal feature number (Binary Classification)

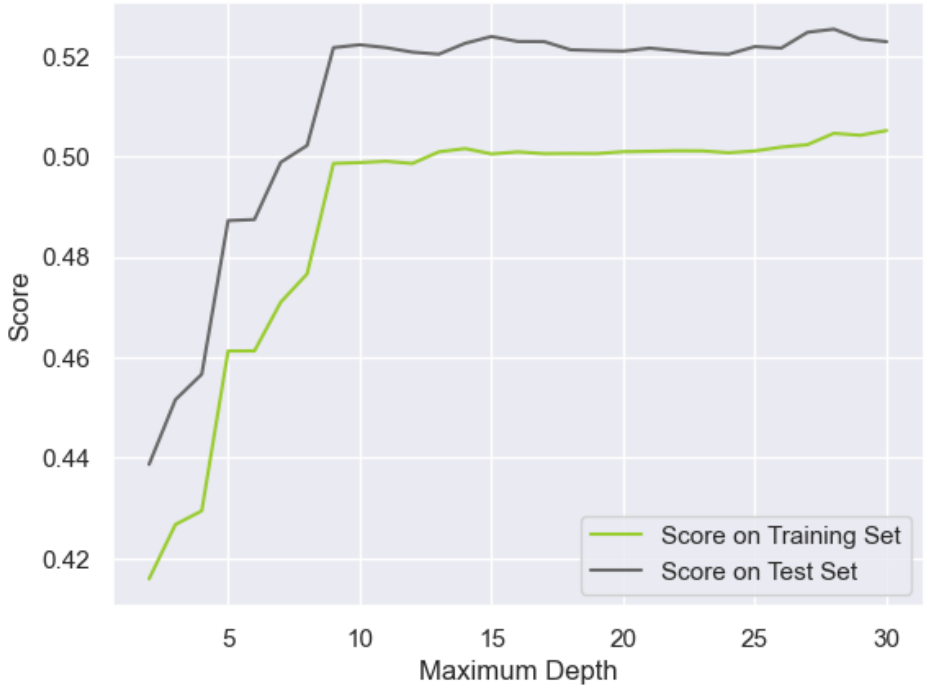

Figure 13: RFE optimal feature number (Multiclass Classification)

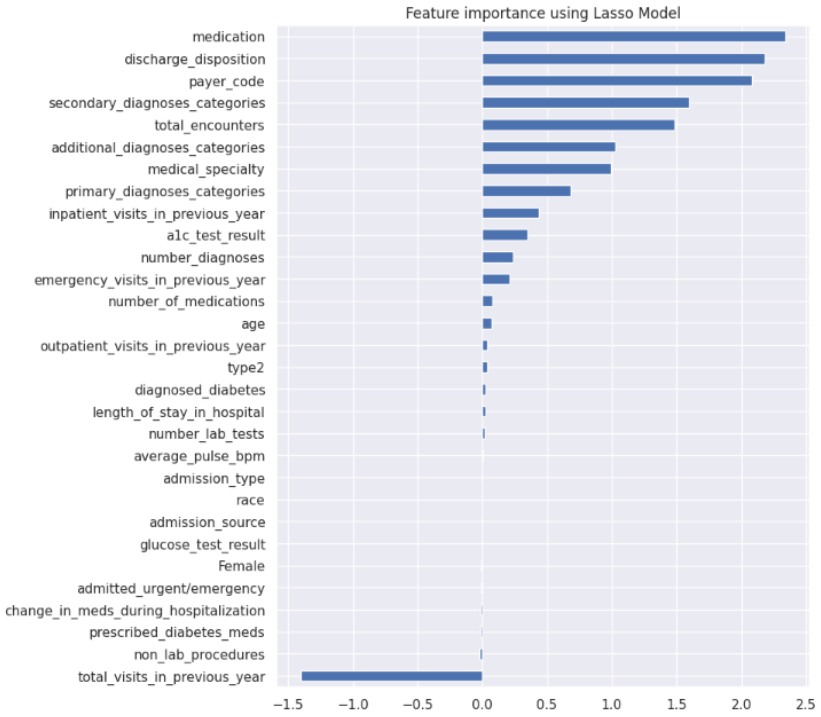

Figure 14: Lasso Regression Results (Binary Classification)

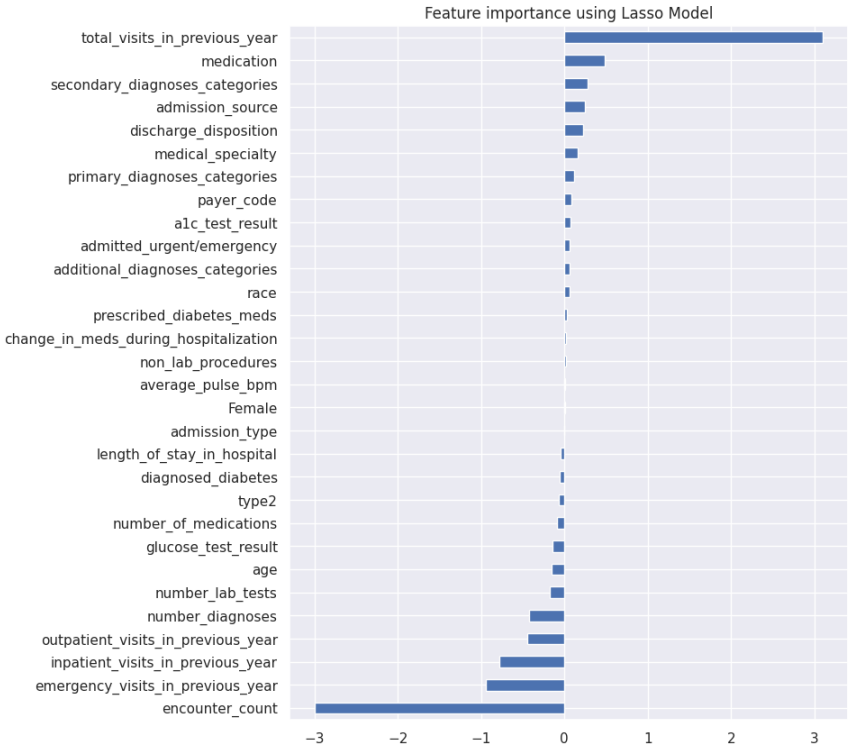

Figure 15: Lasso Regression Results (Multiclass Classification)

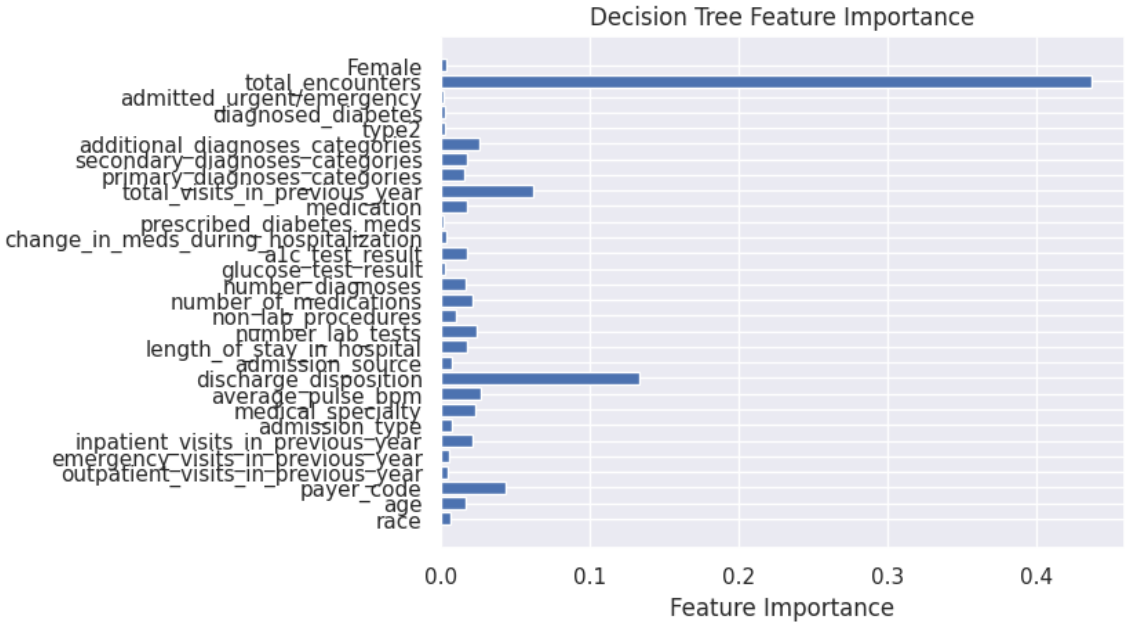

Figure 16: Decision Tree Results (Binary Classification)

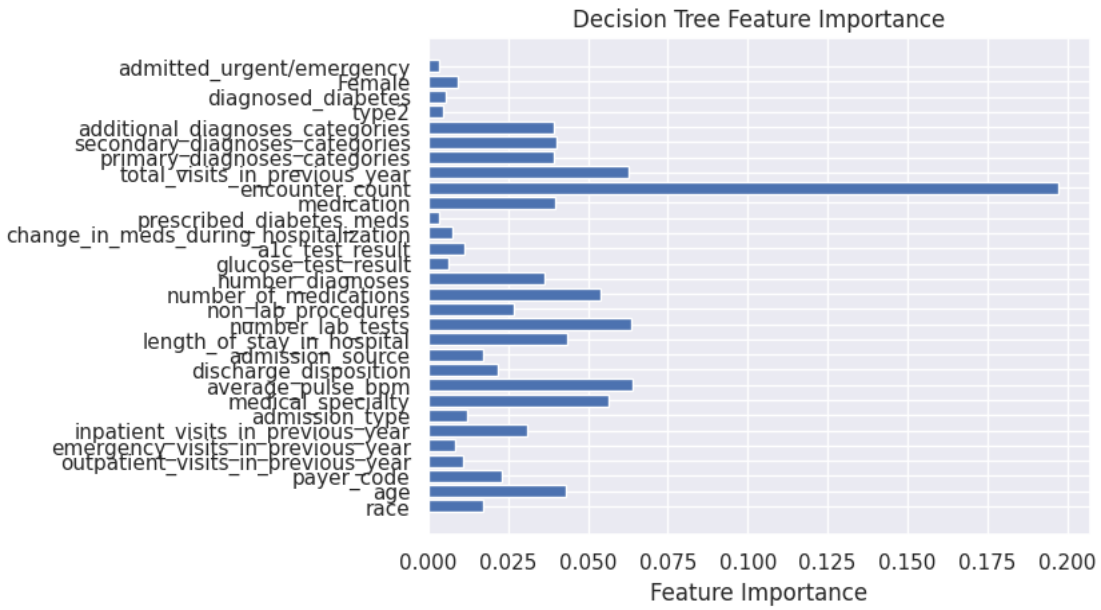

Figure 17: Decision Tree Results (Multiclass Classification)

FEATURES	VARIANCE	SPEARMAN	CHI SQUARE	RFE	LASSO	DECISION TREE	CONCLUSION (min 2 app.)
Country	X	-	-	-	-	-	-
Race	✓	✓	✓	X	X	X	**drop**
Age	✓	X	na	✓	X	✓	keep
Payer_Code	✓	X	✓	X	X	✓	**drop**
Outpatient_Visits_In_Previous_Year	✓	✓	na	X	X	X	**drop**
Emergency_Visits_In_Previous_Year	✓	✓	na	✓	✓	X	keep
Inpatient_Visits_In_Previous_Year	✓	✓	na	✓	✓	✓	keep
Admission_Type	✓	X	✓	X	X	X	**drop**
Medical_Specialty	✓	✓	✓	✓	✓	✓	keep
Average_Pulse_Bpm	✓	X	na	X	X	✓	**drop**
Discharge_Disposition	✓	✓	✓	✓	✓	✓	keep
Admission_Source	✓	X	✓	X	X	X	**drop**
Length_Of_Stay_In_Hospital	✓	✓	na	X	X	✓	keep
Number_Lab_Tests	✓	X	na	X	X	✓	**drop**
Non_Lab_Procedures	✓	X	na	X	X	X	**drop**
Number_Of_Medications	✓	✓	na	✓	X	✓	keep
Number_Diagnoses	✓	X	na	✓	✓	✓	keep
Glucose_Test_Result	✓	✓	✓	X	X	X	**drop**
A1c_Test_Result	✓	✓	✓	X	X	X	**drop**
Change_In_Meds_During_Hospitalization	✓	✓	✓	X	X	X	**drop**
Prescribed_Diabetes_Meds	✓	✓	✓	X	X	X	**drop**
Medication	✓	✓	✓	✓	✓	✓	keep
Total_Visits_In_Previous_Year	✓	X	na	✓	✓	✓	keep
Primary_Diagnoses_Categories	✓	✓	✓	✓	X	✓	keep
Secondary_Diagnoses_Categories	✓	✓	✓	✓	X	✓	keep
Additional_Diagnoses_Categories	✓	✓	✓	X	X	✓	keep
Type2	✓	X	✓	X	X	X	**drop**
Diagnosed_Diabetes	✓	✓	X	X	X	X	**drop**
Admitted_Urgent/Emergency	✓	✓	✓	X	X	X	**drop**
Total_Encounters	✓	✓	na	✓	✓	✓	keep
Female	✓	X	X	X	X	X	**drop**
*na – not applicable							

Figure 18: Feature Selection Overview (Binary Classification)

FEATURES	VARIANCE	SPEARMAN	CHI SQUARE	RFE	LASSO	DECISION TREE	CONCLUSION (min 2 app.)
Country	X	✓	-	-	-	-	-
Race	✓	X	✓	X	X	X	**drop**
Age	✓	✓	na	✓	✓	✓	keep
Payer_Code	✓	✓	✓	X	X	X	**drop**
Outpatient_Visits_In_Previous_Year	✓	✓	na	✓	✓	X	keep
Emergency_Visits_In_Previous_Year	✓	✓	na	✓	✓	X	keep
Inpatient_Visits_In_Previous_Year	✓	✓	na	✓	✓	✓	keep
Admission_Type	✓	X	✓	X	X	X	**drop**
Medical_Specialty	✓	✓	✓	X	X	✓	keep
Average_Pulse_Bpm	✓	X	na	X	X	✓	**drop**
Discharge_Disposition	✓	✓	✓	✓	✓	X	keep
Admission_Source	✓	✓	✓	X	X	X	keep
Length_Of_Stay_In_Hospital	✓	✓	na	X	X	✓	keep
Number_Lab_Tests	✓	✓	na	✓	✓	✓	keep
Non_Lab_Procedures	✓	X	na	X	X	✓	**drop**
Number_Of_Medications	✓	✓	na	X	X	✓	keep
Number_Diagnoses	✓	✓	na	✓	✓	✓	keep
Glucose_Test_Result	✓	X	✓	X	X	X	**drop**
A1c_Test_Result	✓	X	✓	X	X	X	**drop**
Change_In_Meds_During_Hospitalization	✓	X	✓	X	X	X	**drop**
Prescribed_Diabetes_Meds	✓	✓	✓	X	X	X	**drop**
Medication	✓	✓	✓	✓	✓	✓	keep
Total_Visits_In_Previous_Year	X	X	na	✓	✓	✓	keep
Primary_Diagnoses_Categories	✓	✓	✓	X	X	✓	keep
Secondary_Diagnoses_Categories	✓	X	✓	✓	✓	✓	keep
Additional_Diagnoses_Categories	✓	✓	✓	X	X	✓	keep
Type2	✓	✓	✓	X	X	X	**drop**
Diagnosed_Diabetes	✓	X	X	X	X	X	**drop**
Admitted_Urgent/Emergency	✓	X	✓	X	X	X	**drop**
Encounter_Count	✓	✓	na	✓	✓	✓	keep
Female	✓	X	✓	X	X	X	**drop**
*na – not applicable							

Figure 19: Feature Selection Overview (Multiclass Classification)

e. Modeling

Model	Random Search Parameters	Best Parameters
Logistic Regression	'C': [200, 100, 10, 0.1, 0.001]	{'solver': 'newton-cg', 'penalty' None, 'C': 200}
	'solver': ['newton-cg', 'sag', 'saga', 'lbfgs']	
	'penalty': [None, 'l1', 'l2']	
Naive Bayes	'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10,20,30,40,50,60,70,80,90, 100]	{'norm': False, 'fit_prior': True, 'alpha': 0.001}
	'fit_prior': [True, False]	
	'norm': [True, False]	
KNN	'n_neighbors' : range(30, 100, 5)	{'weights': 'distance', 'p': 1, 'n_neighbors': 30, 'metric': 'euclidean', 'algorithm': 'kd_tree'}
	'weights' : ['uniform', 'distance']	
	'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']	
	'metric' : ['euclidean']	
	'p': [1, 2]	

Multi-Layer Percepton	'hidden_layer_sizes': [(100), (100, 50), (100, 75, 50)]	{'solver': 'adam', 'learning_rate_init': 0.001, 'learning_rate': 'invscaling', 'hidden_layer_sizes': (100, 50), 'batch_size': 200, 'activation': 'relu'}
	'activation': ['tanh', 'relu', 'logistic']	
	'activation': ['tanh', 'relu', 'logistic']	
	'solver': ['adam', 'sgd']	
	'learning_rate': ['invscaling', 'adaptive']	
	'batch_size': [50, 100, 200]	
Decision Trees	'splitter': ['best', 'random']	'splitter': 'best', 'min_samples_split': 1000, 'min_samples_leaf': 100, 'max_leaf_nodes': None, 'max_features': 15, 'max_depth': 10, 'criterion': 'gini'
	'max_depth': [9, 10]	
	'min_samples_split': [100, 500, 1000]	
	'min_samples_leaf': [100, 200, 500]	
	'max_features': [None, 15, 0.5]	
	'max_leaf_nodes': [None, 5, 10]	
	'criterion': ['gini', 'entropy']	
Random Forest	'max_depth': [7, 8, 9]	{'min_samples_split': 100, 'min_samples_leaf': 100, 'max_leaf_nodes': None, 'max_features': None, 'max_depth': 9, 'criterion': 'entropy'}
	'min_samples_leaf': [50, 100, 200, 500]	
	'min_samples_split': [50, 100, 200, 500]	
	'max_features': [None, 15, 0.5]	
	'max_leaf_nodes': [None, 5, 10]	
	'criterion': ['gini', 'entropy']	
Support Vector Classifier (SVC)	'kernel': ['linear', 'poly', 'rbf', 'sigmoid']	{'kernel': 'rbf', 'gamma': 0.01, 'C': 50}
	'gamma': [0.001, 0.01, 0.1]	
	"C": [0.001, 0.1, 1, 10, 50, 100]	
Gradient Boosting	'n_estimators': range(40,101,10)	{'learning_rate': 0.2, 'max_depth': 4, 'min_samples_leaf': 50, 'min_samples_split': 500, 'n_estimators': 140, 'subsample': 1.0}
	'learning_rate': [0.05, 0.1, 0.2]	
	'max_depth': [3, 5, 7]	
	'min_samples_split': [2, 5, 10]	
	'min_samples_leaf': [1, 2, 4]	
	'subsample': [0.8, 0.9, 1.0]	

Annex *Table 1: Binary Classification Hyperparameter Tunning with Random Search*

Model	Random Search Parameters	Best Parameters
Logistic Regression	'C': loguniform(1e-4, 1e4)	{'C': 172.79373898388363, 'class_weight': 'balanced', 'max_iter': 100, 'multi_class': 'multinomial', 'n_jobs': -1, 'penalty': None, 'solver': 'sag'}
	'class_weight' : ['balanced']	
	'solver': ['newton-cg', 'sag', 'saga', 'lbfgs'	
	'penalty': [None, 'l2']	
	'n_jobs': [-1]	
	'max_iter': [100, 200, 300]	
	'multi_class': ['multinomial']	
Naive Bayes	'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10,20,30,40,50,60,70,80,90, 100]	{'norm': False, 'fit_prior': True, 'alpha': 0.001}
	'fit_prior': [True, False]	
	'norm': [True, False]	
KNN	'n_neighbors': range(1, 31)	{'weights': 'distance', 'p': 1, 'n_neighbors': 19}
	'weights' : ['uniform', 'distance']	
	'p': [1, 2]	
Multi-Layer Percepton	'hidden_layer_sizes': [(10, 10), (15, 15), (15, 10, 15)]	{'solver': 'adam', 'learning_rate': 'adaptive', 'hidden_layer_sizes': (15, 10, 15), 'batch_size': 50, 'activation': 'tanh'}
	'activation': ['tanh', 'relu', 'logistic']	
	'solver': ['adam', 'sgd']	
	'learning_rate': ['invscaling', 'adaptive']	
	'batch_size': [50, 100, 200]	
Decision Trees	'splitter': ['best', 'random']	'splitter': 'best', 'min_samples_split': 1000, 'min_samples_leaf': 100, 'max_leaf_nodes': None, 'max_features': 15, 'max_depth': 10, 'criterion': 'gini'
	'max_depth': [3,4,5]	
	'min_samples_split': randint(2, 1000)	
	'min_samples_leaf': randint(1, 500)	
	'max_features': [None, 'sqrt', 'log2']	
	'max_leaf_nodes': [None, 5, 10, 20]	
	'criterion': ['gini', 'entropy']	
	'class_weight' : ['balanced']	

Random Forest	'max_depth': [4,5,6]	{'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 6, 'max_features': 15, 'max_leaf_nodes': None, 'min_samples_leaf': 112}
	'min_samples_leaf': randint(100, 501)	
	'class_weight' : ['balanced']	
	'max_features': [None, 15, 0.5]	
	'max_leaf_nodes': [None, 5, 10]	
	'criterion': ['gini', 'entropy']	
Support Vector Classifier (SVC)	'kernel': ['linear', 'rbf']	{'kernel': 'linear', 'gamma': 'scale', 'class_weight': 'balanced'}
	'gamma': ['scale', 'auto']	
	'class_weight':['balanced']	
Gradient Boosting	'n_estimators': randint(40, 101)	{'learning_rate': 0.2, 'max_depth': 4, 'min_samples_leaf': 50, 'min_samples_split': 500, 'n_estimators': 140, 'subsample': 1.0}
	'learning_rate': [0.05, 0.1, 0.2]	
	'max_depth': [4,5,6]	
	'min_samples_split': randint(2, 11)	
	'min_samples_leaf': randint(1, 5)	
	'subsample': [0.8, 0.9, 1.0]	

Annex *Table 2: Multiclass Classification Hyperparameter Tunning with Random Search*

Models	F1-Score K-NN Imputer Approach	F1-Score Mode Approach
Random Forest	f1_train: 0.33 | f1_val: 0.30	f1_train: 0.35 | f1_val: 0.31
Logistic Regression	f1_train: 0.31 | f1_val: 0.28	f1_train: 0.3 | f1_val: 0.28
SVC	f1_train: 0. 3| f1_val: 0.27	f1_train: 0.29| f1_val: 0.32
*** Models performed with class_weights method		

Annex *Table 3: Best 3 Models in Binary Classification - k-NN imputer VS Mode*

Models	F1-Score K-NN Imputer Approach	F1-Score Mode Approach
MLP	f1_train: 0.59 | f1_val: 0.55	f1_train: 0.63 |f1_val: 0.29
Random Forest	f1_train: 0.58 | f1_val: 0.55	f1_train: 0.58 | f1_val: 0.53
Decision Trees	f1_train: 0.57 | f1_val: 0.54	f1_train: 0.83 | f1_val: 0.48

Annex *Table 4: Best 3 Models in Multiclass Classification - k-NN imputer VS Mode*

Model	Imbalance Class Approach	Binary	Multiclass
Logistic Regression	SMOTE	f1_train: 0.64 f1_val: 0.28	f1_train: 0.50 f1_val: 0.52
	class_weights = 'balanced'	f1_train: 0.31 f1_val: 0.28	f1_train: 0.58 f1_val: 0.53
Complement Naive Bayes	SMOTE	f1_ train: 0.60 f1_val: 0.27	f1_ train: 0.40 f1_val: 0.44
	***	f1_ train: 0.29 f1_val: 0.27	f1_ train: 0.54 f1_val: 0.53
KNN	SMOTE	f1_train: 1.00 f1_val: 0.25	f1_train: 1.00 f1_val: 0.51
MLP	SMOTE	f1_train: 0.84 f1_val: 0.21	f1_train: 0.59 f1_val: 0.55
Decision Trees	SMOTE	f1_train: 0.87 f1_val: 0.10	f1_train: 0.58 f1_val: 0.52
	class_weights = 'balanced'	f1_train: 0.33 f1_val: 0.29	f1_train: 0.57 f1_val: 0.54
Random Forest	SMOTE	f1_train: 0.86 f1_val: 0.08	f1_train: 0.58 f1_val: 0.55
	class_weights = 'balanced'	f1_train: 0.33 f1_val: 0.30	f1_train: 0.60 f1_val: 0.55
SVC	SMOTE	f1_train: 0.67 f1_val: 0.27	f1_train: 0.46 f1_val: 0.53
	class_weights = 'balanced'	f1_train: 0.30 f1_val: 0.27	f1_train: 0.60 f1_val: 0.53
Gradient Boosting	SMOTE	f1_train: 0.93 f1_val: 0.03	f1_train: 0.73 f1_val: 0.56
- 3 Best Models			
*** **Complement Naive Bayes** does not have the parameter class_weights. However, it is particularly effective when dealing with imbalanced datasets, hence we choose to use it with the dataset X_train (imbalance dataset).			

Annex *Table 5: Binary and Multiclass Models' F1-Scores*

		Random Forest	MLP
F1-Score Weighted	Train	0.58	0.59
	Validation	0.55	0.55
F1-Score (Class 0) Minority Class	Train	0.34	0.64
	Validation	0.28	0.18
F1-Score (Class 1)	Train	0.41	0.42
	Validation	0.36	0.41
F1-Score (Class 2) Majority Class	Train	0.75	0.72
	Validation	0.73	0.74

Annex Table 6: F1-Scores for Random Forest and MLP (Multiclass Classification)