MACHINE LEARNING 2023/2024

READY TO BE DISCHARGED: EXAMINING HOSPITAL READMISSIONS

PROFESSORS ROBERTO HENRIQUES AND RICARDO SANTOS

GROUP 15:
EMÍLIA SANTOS | N.º 20230446
FRANCISCO CASTRO | N.º 20230992
MARIA CARLOTA FORTUNATO | N.º 20230382
INÊS VALVERDE | N.º R20201472

Abstract

This project is based on a challenge the healthcare sector faces nowadays, hospital readmissions. Overcoming this issue is essential since it would result in an overall reduction of healthcare costs, improved ability to allocate resources to patients, and, consequently, improved patient care. Another topic that should be taken into account, since it could affect the quality of healthcare, is the prediction of hospital readmissions.

Our main hypotheses are to predict if a patient will be readmitted or not to the hospital within 30 days of being discharged, through binary classification, and to predict if the patient will be readmitted or not in more or less than 30 days, using multiclass classification.

To conduct these predictions, we started by understanding the problem through literature review. Then, we explored the dataset and preprocessed the data, searching for duplicates, treating missing values and outliers, and other important steps. Afterwards, we dealt with the class imbalance existent in the dataset and conducted feature selection. Finally, we built the predictive models for binary classification and multiclass classification using different classifiers: Logistic Regression, Neural Networks, Naïve Bayes, K-Nearest Neighbours, Decision Trees, Random Forest, and Support Vector Machines. To compare these, we used the F1 score. Another important factor discussed, due to its effects on the models' performances, is the overfitting in our data which we tried to overcome.

The study developed showed that for binary classification the best model is the Random Forest. For the multiclass classification, using random undersampling and random oversampling, the Neural Network was considered. Both of these have shown the best ability to deal with the complex nature of the problem and our data.

Keywords: hospital readmissions; binary classification; multiclass classification; F1 score

Contents

Table of Figures

1. Introduction

Hospital readmissions represent a "situation where a patient was discharged from the hospital and winds up back for the same or related care within 30 or more days". The number of hospital readmissions is often a measure of the quality of hospital care. If it is high, it can mean that the follow-up care was not properly done or that the patient was not fully treated during his/her stay at the hospital (Hospital Readmissions - Glossary, n.d.). Additionally, this represents a significant cost for the hospitals. The Agency for Healthcare Research and Quality (AHRQ) has reported that the average cost for patients readmitted within 30 days of discharge is 41.3 billion dollars (data related to 2011). These are the main reasons why this is an indicator of greater importance and research by the medical industry. In fact, the United States implemented the Hospital Readmission Reduction Program (HRRP) that penalizes hospitals with higher readmission rates than the average.

The main goal of our project is to accurately predict hospital readmissions, using Machine Learning (ML) algorithms. To do this, we will build two predictive models. One using binary classification: To predict if a patient will be readmitted to the hospital within 30 days of being discharged. And another using multiclass classification: To predict if the patient won't be readmitted, readmitted in less than 30 days or more than 30 days.

An accurate prediction of hospital readmissions improves the quality of the healthcare system and reduces costs. Moreover, it can provide timely diagnosis and effective treatment for patients, prevent complications due to delayed or incorrect treatment and shorten hospitalization time.

Taking a look at other similar researches, there are several reviews with respect to predicting readmission with Machine Learning techniques. According to Huang, Y., Talwar, A., Chatterjee, S., & Aparasu, R. R. (2021), which analyzes 43 studies involving Machine Learning prediction in hospital readmissions, all of them used a binary readmission outcome. Most of the studies considered only one type of readmission rate and the most popular was a 30-day readmission.

The most-used ML techniques were Tree-based methods, such as Decision Trees and Random Forests, followed by Neural Networks and regularized Logistic Regression. Regarding the model performance, it is noted a variation in terms of AUC (area under ROC curve) across predictive models but most of them performed an AUC above 0.7, meaning the models have a good discriminatory ability. It is to note that the main challenges for achieving a model with high performance is the use of rich information due to the multidimensional nature of the problem and the dependence of Machine Learning techniques on the quality and information of input data.

2. Data Exploration

We started by exploring and preprocessing our data. Our first approach was based on understanding and searching for previous literature, which contributed to our business understanding. Afterwards, we began the data exploration by analyzing the dataset and its variables and looking for possible errors, as well as outliers, incoherencies, and missing values.

We were given a CSV file with 71 236 rows and 31 columns, from which 11 of the variables are numerical and 20 of them are categorical (Annex 1).

According to the **descriptive statistics** (Annex 2 and Annex 3), it seems that there are no missing values in the numerical variables and that there are missing values in most all categorical

variables. Focusing on the **numeric variables** (Annex 2), the information that we extract is the possible existence of outliers in the following features: *outpatient_visits_in_previous_year, emergency_visits_in_previous_year*, *inpatient_visits_in_previous_year,* and *number_of_medications*, since the maximum value differs very much from the quantiles. For instance, the variable *outpatient_visits_in_previous_year* has a maximum value of 42 and quartile values equal to 0, which suggests that it is likely to have outliers. Regarding the **categorical variables** (Annex 3), the column *country* only contains the value "USA". Given this, the variable can be excluded from our dataset (as we later did), according to the filter feature selection method as it doesn't contribute significantly and might potentially interfere with the results. Looking into further special cases found in the analysis, we can explore the following:

Variables	Values to consider	Meaning	How to deal/treat
weight, race	'?'	Missing value	Replaced by NaN
payer_code	'?'	Without insurance or insurance provider unknown	Replaced by 'No insurance/Unknown'
medical_specialty	'?'	Other medical specialty	Replaced by 'Unknown'
gender	'Unknown/Invalid'	Missing value	Replaced by NaN
medication	'[]'	Individuals without medication prescribed during the encounter	Replaced by 'No Medication' for better understanding
admission_type, admission_source	'Not Mapped'	Information that was not collected	Replaced by NaN

Table 1 - Data exploration of some categorical variables

Visualizations

To further explore our data, some visualizations were made. Countplots allowed us to understand the distribution of some features (Annex 4). Our train dataset is composed of approximately 54% of females and 46% of males (Annex 4.1). About 30% of the patients are between 70 and 80 years old (Annex 4.2). Most of the individuals (77%) were on diabetes medications during their encounter (Annex 4.3). Plus, the most common primary diagnosis (Annex 4.4) is related to circulatory system diseases (30.08%). Most of the patients stayed for 3 days in the hospital, but the longest stay was 14 days (Annex 4.5).

The dataset has 71 236 hospital admissions and 53 985 unique patients. Approximately, 16% of patients have multiple admissions and the maximum number of admissions for a patient is 18. Patients were administered on average 15 medications during the encounter, and the average number of lab procedures is 42.

It is also noticed that some variables have a high percentage of the same values, namely, 78% of 'caucasian' in *race*, 77% of ones in *prescribed_diabetes_meds*, 83% of zeros in *outpatient_visits_in_previous_year*, 89% zeros in *emergency_visits_in_previous_year* and 66% zeros in *inpatient_visits_in_previous_year*. However, we will proceed with them and wait until the feature selection.

We've also made pairwise relationships (Annex 5) that don't show any significant relationship between variables. However, a relationship between *age* and *number_of_medications* is captured: higher the age, higher the number of medications (Annex 6).

3. Data Preprocessing

3.1. Duplicates

Duplicates are considered entries that have the same values, they transmit the same information.

Looking for duplicated entries in our dataset, we first find two different indexes: *encounter_id* and *patient_id*. We know that *encounter_id* is a variable that represents each visit to the hospital. Every time a patient goes to the hospital, there will be a different *encounter_id*. If this value appears duplicated, something is wrong but there was none. Afterwards, we transformed *encounter_id* into the index of our data and searched for identical rows because there could be cases where *encounter_id* was not duplicated, but hospital visits might be recorded more than once. However, this was not the case. In the case of *patient_id*, there are duplicated entries which means that the same patient visited the hospital several times. Still, this is not useful in predictive terms and this variable will be removed.

3.2. Missing Values

Missing values represent the absence of information, which means that no value was stored for a particular feature in the dataset. If these values are not correctly handled, we can end up with inaccurate results. To address this various techniques can be used, namely, replacing with constant values, measures of central tendency (in this case with categorical features, the mode), and KNN imputation. Below, we discuss which ones were used.

It was possible to spot missing values in the following variables: *race* (7.06%), *age* (4.97%), *admission_type* (5.50%), *discharge_disposition* (4.55%), *admission_source* (6.85%), *primary_diagnosis* (0.02%), *secondary_diagnosis* (0.38%), and *additional_diagnosis* (1.46%), *weight* (96.87%), *glucose_test_result* (94.87%) and *a1c_test_result* (83.27%).

In fact, the missing values in *glucose_test_result* and *a1c_test_result* represent cases where these tests were not made to the patients. As it can provide meaningful insights, these will be replaced by "Not Measured". Also, in the variables *admission_type* and *admission_source* missing values were replaced by "Not Available". According to Noridian Healthcare Solution (2023), "Not Available" is a category used in these features. In the case of *secondary_diagnosis* and *additional_diagnosis,* they mean that the patient doesn't have a diagnosis, so they will be replaced by "No Diagnosis".

Since the feature *weight* has a high percentage of missing values, it will be removed because it does not contribute to the predictions, and imputing those values would introduce bias in our model.

After testing several methods (mode imputation and KNN with 2, 3, and 5 neighbors) for the remaining features, KNN imputation (with 2 neighbors) would be the best option. KNN imputer provides more accurate imputations rather than mode imputation.

3.3. Outliers

Outliers are values that differ very much from the others and whose existence can lead to misinterpretations, due to its effects on other metrics and algorithms, such as the mean.

To better identify them, we analyzed the numeric variables boxplots and histograms (Annex 7), and the variables that caught our attention the most due to the high amount of outliers, were *outpatient_visits_in_previous_year, inpatient_visits_in_previous_year, emergency_visits_in_previous_year, number_lab_tests,* and *number_of_medications*. To treat the outliers, the methods Interquartile Range (IQR) and manually defining thresholds were used. After these, we decided to use the combination of both these methods, since doing this would be beneficial for our data integrity. Doing the manual method, the percentage of data kept after removing outliers was 98.49%, using the IQR method this percentage decreased to 65.93% (too much information loss), and with the combination of both, the percentage corresponds to 99.4% (less loss of information).

3.4. Incoherencies

The first incoherency that we came across was related to the variable *discharge_disposition*. This feature has values that correspond to expired patients, meaning that they eventually deceased. These individuals will never be readmitted, so it is not correct to train our model with those observations - they will be removed from train. The approach in the validation and test datasets is to store those observations apart from the others, give the value "No" to the target variables, and in the final model we will concatenate those patients on the final predictions. The following table explains the incoherencies based on the values of the features *primary_diagnosis, secondary_diagnosis,* and *additional_diagnosis.* In our understanding, the same patient cannot have the same diagnosis in the secondary and additional diagnosis. In these variables, some values are not in a 3-digit code format, but this is overcome in feature engineering.

Conditions	Replacements
primary_diagnosis being the same as *secondary_diagnosis;* if the value of *additional_diagnosis* is different	Replace *additional_diagnosis* by 'No diagnosis' and use the previous value in *secondary_diagnosis*
secondary_diagnosis being the same as *additional_diagnosis*; *primary_diagnosis* being equal to *additional_diagnosis*	Replace by 'No diagnosis' in *additional_diagnosis*
primary_diagnosis, *secondary_diagnosis*, and *additional_diagnosis* being the same	Replace *secondary_diagnosis* and *additional _diagnosis* by 'No diagnosis'

Table 2 - Incoherencies on *primary_diagnosis, secundary_diagnosis and additional_diagnosis*

3.5. Feature Engineering

Feature Engineering consists of the creation, transformation, extraction, and manipulation of variables. Often, in predictive models with poor performance can be explained by the way the variables are represented, avoiding the models to learn effectively.

Various features were modified and some were created based on others, essentially to reduce the number of categories on some features and to find better representations of the predictors (Butcher & Smith, 2020). In this phase, we have created the variables *encounter_count, age_groups, has_insurance,* and *type_II_diabetes*. Additionally, the variables *caucasian,*

medical_specialty and *primary_diagnosis, secondary_diagnosis,* and *additional_diagnosis* were transformed by grouping categories (Annex 8).

3.6. Encoding & Data Scaling

As most of the algorithms implemented require numerical inputs, it is needed to transform non-metric features into metric. This can be achieved through encoding. There are several methods to do it, we started by transforming into binary the variables: *prescribed_diabetes_meds*, where the "Yes" values were replaced by 1 and "No" by 0*, change_in_meds_during_hospitalization,* where 0 is "No" and 1 is "Ch'" *readmitted_binary*, 0 is "No" and 1 is "Yes" and, *gender* where 0 is "Male" and 1 is "Female".

Afterwards, we implemented the ordinal encoding (Self Study 11.1.1) to the variables *age*, *glucose_test_result, a1c_test_result,* and *age_groups,* since there is an order among its categories. Then, the target encoding (Self Study 11.1.2) was applied to the variable *payer_code, admission_type, discharge_disposition, admission_source, primary_diagnosis, secondary_diagnosis, additional_diagnosis, medication* and *medical_specialty*. These variables were target encoded due to its high number of categories and by doing so, we are avoiding the curse of dimensionality, which could have happened if we used other encodings such as the one-hot encoding. For the *readmitted_multiclass* variable, label encoding (Self Study 11.1.3). In this case, "<30", ">30", and "No" are transformed into 0, 1, and 2, respectively.

Regarding the scaling of our data, we decided to use Min-Max scaling to our metric features, ordinal and target encoded variables, transforming these features by normalizing their values between 0 and 1.

4. Imbalanced Data

In binary classification, the target variable is composed of about 88% of 0s ("No") and 11% of 1s ("Yes"), whereas in multiclass classification the target variable has 11% of 0s ("<30"), 36% of 1s (">30") and 53% of 2s ("No"). This is known as imbalanced data which is a challenge that will impact the performance of classifiers - they will perform well on the majority class but poor on the minority class(es), given the influence of the larger majority class (Ganganwar, V., 2012).

There are several approaches to address this, namely data-level methods that include resampling techniques (such as oversampling and undersampling), algorithm-level such as hyperparameter tuning and the use of ensemble methods (e.g. Random Forest), or even hybrid methods that combine both (Krawczyk, 2016). In order to explore different methods, we created several datasets (Annex 9). We have implemented random undersampling, random undersampling, and random oversampling, SMOTE-NC and SMOTE-ENN. These techniques are presented in the Self Study 11.2.

5. Binary Classification

In binary classification, we will use the variable *readmitted_binary* as our target variable, i.e. the variable we want to predict. It is important to note that the variable *readmitted_multiclass* cannot be used to predict this binary variable. As a common practice, we implemented the holdout method for the evaluation of each model meaning that the initial dataset was divided into 70% for train and 30% for validation, using the method *train_test_split*. The training set is used to train the model and its performance is assessed on the validation.

From our analysis (Annex 9), we concluded that all of them have similar performances but SMOTE-ENN performed generally worst, followed by SMOTE-NC. Then, random undersampling and both random undersampling and random oversampling performed better, producing similar effects. The main difference found was the number of observations on the resampled dataset, i.e. with undersampling the training dataset has 11 072 observations whereas with undersampling and oversampling the number increased to 89 462. As it is expected, a smaller dataset is computationally more efficient, allowing us to explore different algorithms. This said we proceeded with random undersampling.

5.1. Feature Selection

This phase is aimed at minimizing the problem of excessive and irrelevant features, increasing learning efficiency. Our approach was to apply seven techniques to get a robust analysis of the features' importance.

Starting with the **Filter Methods**, we looked for **Univariate Variables** analysis to examine the variability of numerical features. There were not found features with low variance. Then in the **Spearman Correlation** (Annex 10) there is a high correlation between *total_visits_in_previous_year* and *inpatient_visits_in_previous_year* (0.8)*, emergency_visits_in_previous_year* (0.5) and *outpatient_visits_in_previous_year* (0.5), showing redundancy and *average_pulse_bpm* shows to be irrelevant (low correlation). After that, the **Chi-Square Test of Independence** which tests the independence between each categorical variable and the target variable, stated that 5 features are not important predictors. Finally, **SelectKBest** using Mutual Information (Self Study 11.3) selected the best 10 features based on the highest mutual information they have with the target variable.

Moving to **Wrapper Methods**, our team applied the **Recursive Feature Elimination (RFE)** using Logistic Regression that determined the optimal number of features removing insignificant features at a time until it reaches the best ones according to their contribution to the model's performance. The optimal number of features to retain is 3.

Next, on **Embedded Methods**, **LASSO Regression** is a popular one that shrinks some of the coefficients to zero. Non-zero coefficients indicate that the corresponding features are considered important by the model, in this case, only one feature to eliminate. **Random Forest** is also a good method as it combines the predictions of multiple decision trees. The importance of each feature is determined by how much it contributed to make decisions in the training. The variables to keep here were 12.

The summary of our findings can be found in Annex 11. The final decision was to hold the features that retain more than 50% of the previous methods: *age, inpatient_visits_in_previous_year, average_pulse_bpm, discharge_disposition, length_of_stay_in_hospital, number_lab_tests, non_lab_procedures, number_of_medications, primary_diagnosis, secondary_diagnosis, additional_diagnosis, medication, total_visits_in_previous_year,* and *encounter_count.*

5.2. Modelling

After the preprocessing and resampling of the dataset, classification models were applied and we explored hyperparameter tuning in each model, using Random Search and Grid Search (Self Study 11.4). Additionally, we explored the needed parameters individually to be able to know which parameters to input in those hyperparameter tuning methods. The evaluation of each model was

made using AUC (Annex 15) and F1 score which is the harmonic average between precision and recall, being the most appropriate measure given the imbalanced dataset.

Following the literature, we implemented common algorithms used namely Naïve Bayes, Logistic Regression, Decision Trees, K-Nearest Neighbors, Neural Networks, Random Forest, and Support Vector Machines (SVM). In a comparative study for hospital readmissions, Alajmani, S. & Elazhary, H. (2019) showed that SVM had an improved performance, whereas Bayes Classifier and Logistic Regression were not so promising. Tree-based methods are also found to be good classifiers. In our research, we will explore in detail Logistic Regression, Decision Trees, and Random Forest and Support Vector Machines in order to understand what makes them outperform others and, in the results, the conclusions regarding all models will be presented.

In **Logistic Regression**, when exploring solutions to address the class imbalance, we found that the parameter *class_weight* set to "balanced" improves our model significantly. This mode automatically adjusts the weights of the input features inversely proportional to the class frequencies, using the values from the target variable. This said we ended up having a better model (with reduced overfitting) by not using the resampled dataset. The best scores achieved in train and validation are **0.31** and **0.29**, respectively.

Decision Trees have several advantages such as easy interpretation, insensitivity to the scale of the variables, and capacity to deal with different types of data. On the other hand, this classifier is prone to overfitting due to the complexity of the trees. Once again, we found significantly better results using adjusted class weights in the dataset that wasn't resampled. The best model using the resampled dataset has 0.62 (train) and 0.27 (validation), whereas the best model making use of the parameter previously mentioned performed **0.31** on train and **0.29** on validation. We also fixed the minimum number of samples required at a leaf node and the maximum depth of the tree, which can control overfitting.

Random Forest is widely found in previous literature about this topic and performs well. It is a special case of bagging ensemble: multiple decision trees are created and each one is trained on a random subset of the training set (also known as bootstraps) and then for each one the predictions are calculated, returning the output based on the majority voting. We started by defining the parameters we wanted to explore: *n_estimators*, *max_samples*, *max_depth,* and *class_weight*. Resorting to Grid Search, we were able to achieve the best scores for this model being **0.31** and **0.30** in train and validation, respectively.

Support Vector Machines aims to perform classification by finding the hyper-plane that differentiates the two classes very well. In the binary classification, the best hyperplane is the one with the largest margin between the classes. We decided to explore the *C*, *kernel*, *degree, gamma,* and *class_weight* parameters through several tries using Random Search and Grid Search. The final model was improved to **0.32** on train and **0.29** on validation. This model doesn't use the resampled dataset, but the balanced class weight.

5.3. Results and Main Findings

Starting with Bayes Classifier, when exploring the parameters we found **Gaussian Naïve Bayes** and **Complement Naïve Bayes** (Self Study 11.5), which is particularly suited for imbalanced data. Since these classifiers work under strong assumptions, being one of them that the attributes

follow a Gaussian distribution and we are not able to know the distribution of our data, these were used only for exploration, not being the best option.

In **K-Nearest Neighbors** and **Neural Networks**, after optimizing the parameters, we obtained a significantly higher overfitting and poor performance compared to others. The main difference is that these two don't afford a parameter that handles the class weight, so we used the resampled dataset. In our research, the models that can adjust the class weights have reduced overfitting and better performance.

Finally, we consider **Random Forest** as the best predictive model. This classifier is able to capture non-linear relationships between variables and complex relationships and, in the case of hospital readmissions, those are often found since it involves considering different factors from different natures (demographics, health, social). When compared to Decision Trees, it also performed well. Random Forest tends to be more robust, in the sense that controls better overfitting because it combines the predictions from multiple decision trees and includes parameters for this purpose. This model scored 0.3082 in Kaggle, also the most successful one.

Gaussian Naïve Bayes	Complement Naïve Bayes	Logistic Regression	Decision Trees
Train: 0.480	Train: 0.276	Train: 0.313	Train: 0.314
Validation: 0.267	Validation: 0.267	Validation: 0.286	Validation: 0.289
Difference: 0.213	Difference: 0.008	Difference: 0.027	Difference: 0.024
K-Nearest Neighbors	**Neural Networks**	**Support Vector Machines**	**Random Forest**
Train: 0.620	Train: 0.591	Train: 0.327	Train: 0.315
Validation: 0.257	Validation: 0.242	Validation: 0.295	Validation: 0.302
Difference: 0.362	Difference: 0.349	Difference: 0.032	Difference: 0.013

Table 3 - Model Performance for Binary Classification (Final Dataset)

6. Multiclass Classification

Having a multiclass classification problem, we have to use the variable *readmitted_multiclass* as our target variable. Moreover, the variable *readmitted_binary* will not be included in the model.

As we already talked about, we tested different sampling methods (Annex 14), in this case, we chose the combination of random undersampling and random oversampling and with it, we achieved a better performance than with other resampling methods. This might have occurred due to the balance between classes that this method provides, allowing the model to learn equally from all classes. For the model evaluation, we employed the weighted F1 score, which differs from the one used in a binary classification.

6.1. Feature Selection

For the multiclass classification, the same feature selection techniques were applied. Univariate Variables showed that all variables were relevant. Next, the Spearman Correlation revealed one variable as redundant (*total_visits_in_previous_year*) and another as irrelevant (*average_pulse_bpm*). Chi-Squared Test of Independence didn't identify variables that were not important. Next, Recursive Feature Elimination (RFE) suggested that the optimal number of features is 28, meaning that we only should discard one, *change_in_meds_during_hospitalization*.

LASSO suggested removing also one feature: *outpatient_visits_in_previous_year*. However, there are some variables that have a coefficient near zero and after further investigation, we realized that in *admission_source* and *change_in_meds_during_hospitalization* the low coefficient values mean that the variables are not providing important information as well (Annex 12). Random Forest advised us to exclude 17 features. The last method, KBest, revealed 20 non-essential features.

With these results in mind, we decided to use at least more than 50% as the threshold to keep the variables considered important (Annex 13). The variables excluded were: *outpatient_visits_in_previous_year* and *change_in_meds_during_hospitalization*.

6.2. Modelling

In 2020, Hu implemented Naïve Bayes, Gradient Boosting, Random Forest, Decision Tree, Logistic Regression, and Support Vector Machines (SVM) in a study using multiclass classification of medical data. It is concluded that Gradient Boosting and Random Forest have improved accuracy. Our team explored the following predictive models: Logistic Regression, Multinomial Naïve Bayes, Decision trees, K-Nearest Neighbors, Neural Networks, Random Forest, and Support Vector Machine (SVM). In order to optimize the parameters, we used Random Search and Grid Search.

6.3. Results and Main Findings

Regarding the classifiers specifically used, **Support Vector Machines** seems to be consistent in terms of performance among all datasets (Annex 14), having higher scores and less overfitting. Using the data before without any resampling technique it was obtained a performance of **0.58** and **0.53** for train and validation, achieving the best model for SVM. However, we were limited by Grid Search and Random Search. These are computationally exhaustive, which made it impossible for us to run them in this model. We tried to test the best combinations of parameters, but it proved to be inefficient.

Neural Networks also perform well with insignificant overfitting/underfitting, namely **0.55** in train and **0.57** in validation. The fact that both of these models are able to uncover complex relationships and to handle high-dimensional datasets (i.e. datasets with many features and from different dimensions) are the main reasons for them to be able to predict accurately the multiclass feature. Other models showing not to be so great but also good classifiers as well are **K-Nearest Neighbors** and **Random Forest**. Moreover, **Naïve Bayes**, **Logistic Regression,** and **Decision Trees** are performing worse than others due to increased underfitting. This may reveal a lack of capacity to deal with the complexity of this data since all of these models are simpler than others.

Finally, Neural Networks is considered the best predictive model for multiclass classification due to their great ability to handle the complexity of our data and the fact that it learns gradually according to levels of complexity, making the model more robust.

Logistic Regression	Multinomial Naive Bayes	Decision Tree	K-Nearest Neighbors
Train: 0.490	Train: 0.458	Train: 0.466	Train: 0.575
Validation: 0.527	Validation: 0.508	Validation: 0.512	Validation: 0.502
Difference: -0.038	Difference: -0.050	Difference: -0.047	Difference: 0.073
Neural Networks	**Random Forest**	**Support Vector Machines**	
Train: 0.546	Train: 0.498	Train: 0.583	
Validation: 0.564	Validation: 0.526	Validation 0.525	
Difference: -0.018	Difference: -0.028	Difference: 0.058	

Table 4 - Model Performance for Multiclass Classification (Final dataset)

7. Limitations

Besides the problem of imbalanced data addressed in Section 5, we still face two main problems: low F1 scores and overfitting, which means the model is performing poorly and resulting in overly optimistic estimates during train and doesn't generalize well to new unseen data. This means that the predictive models tend to memorize all the data, including noise on the training set,

instead of learning the discipline hidden behind the data (Xue, 2019). According to Xue (2019), the training set can include noise when the training data is too small in size (which may happen in undersampling), or too many noises (that can happen in oversampling). Since the overfitting significantly reduces when making use of the balanced class weight, we believe that these could be the main reasons. Moreover, as the author suggests, we could have explored "early stopping" strategy and regularization-based algorithms that were only done in Logistic Regression using L1 regularization and L2 regularization.

Another limitation found is that Grid Search is computationally expensive, which limited us on the parameter exploration, especially in the most complex models. Most of the time, we were not able to explore as many parameters as we wanted and, in the case of Support Vector Machines, we were not able to run for the multiclass classification.

8. Conclusion

For the binary classification problem, we concluded that the models that obtained better performances and less overfitting are the ones that don't use the resampled dataset, but instead an algorithm-level method such as the use of the parameter *class_weight*. This was not expected since the use of resampling techniques is often found in other studies when the problem of imbalanced data arises. Moreover, we were not able to increase the F1 score results on the validation dataset to more than around 0.3, which indicates a relatively low performance. Although there are several techniques and methods used to address the challenge of imbalanced data, it will always reflect on the results. Miriam Santos (2022) explored the effect of imbalanced data and the use of oversampling techniques and emphasizes that these imperfections are responsible for the degradation of the algorithm performance and biased predictions.

Another conclusion found was that the performance of the algorithms in the multiclass classification outperforms the performance of the binary classification algorithms. The main reason is the use of a different performance metric: in multiclass classification, the F1 score takes into account the number of samples in each class from the target variable whereas in the binary classification, only the positive class (i.e. 1s) is considered. Additionally, another hypothesis is that the multiclass classification involves more classes, providing the model with richer information. This might allow the model to reveal more patterns in the data, improving the performance.

Our team was also able to see the impact of the preprocessing in the models, emphasizing the importance of this step. For example, after the creation of the variable *encounter_count,* our performance on Kaggle went from 0.26 to 0.3.

As additional future work, other resampling techniques and classification algorithms could be included in the analysis. Moreover, it seems that the multiclass classification in terms of hospital readmissions is not so well-explored yet since we found few studies. Further researches can take that direction.

9. References

[1] 1.10. Decision Trees. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/tree.html

[2] Alajmani, S., & Elazhary, H. (2019). Hospital readmission prediction using machine learning techniques. International Journal of Advanced Computer Science and Applications, 10(4).

[3] Association of American Medical Colleges. (n.d.). Specialty Profiles | Careers in Medicine. Retrieved from https://www.aamc.org/cim/explore-options/specialty-profiles

[4] Automated Machine Learning Methods, Systems, Challenges. Available at: https://library.oapen.org/bitstream/handle/20.500.12657/23012/1/1007149.pdf

[5] Bhandari, P. (2023, June 21). Missing data | Types, explanation, & imputation. Scribbr. https://www.scribbr.com/statistics/missing-data/

[6] Butcher, B., & Smith, B. J. (2020). Feature Engineering and Selection: A Practical approach for predictive models. The American Statistician, 74(3), 308–309. https://doi.org/10.1080/00031305.2020.1790217

[7] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over–sampling technique. Journal of Artificial Intelligent Research,16, 321–357.

[8] GeeksforGeeks. (2023, April 10). Complement Naive Bayes CNB algorithm. https://www.geeksforgeeks.org/complement-naive-bayes-cnb-algorithm/

[9] Hines, A. L., Barrett, M. L., Joanna, H., Jiang, P. D., & Steiner, C. A. (2014). Statistical brief# 172. Agency for Healthcare Research and Quality.

[10] Hospital readmissions - glossary. (n.d.). HealthCare.gov. https://www.healthcare.gov/glossary/hospital-readmissions/

[11] Hu, Y. (2020, December 26). Explainable multi-class classification of medical data. arXiv.org. https://arxiv.org/abs/2012.13796

[12] 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

[13] J. Prusa, T. M. Khoshgoftaar, D. J. Dittman and A. Napolitano, "Using Random Undersampling to Alleviate Class Imbalance on Tweet Sentiment Data," 2015 IEEE International Conference on Information Reuse and Integration, San Francisco, CA, USA, 2015, pp. 197-202, doi: 10.1109/IRI.2015.39.

[14] Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence, 5(4), 221–232. https://doi.org/10.1007/s13748-016-0094-0

[15] Michailidis, P., Dimitriadou, A., Papadimitriou, T., & Gogas, P. (2022). Forecasting Hospital Readmissions with Machine Learning. Healthcare, 10(6), 981. https://doi.org/10.3390/healthcare10060981

[16] Muntasir Nishat, M. et al. (2022) A comprehensive investigation of the performances of different machine learning classifiers with smote-enn oversampling technique and hyperparameter optimization for Imbalanced Heart Failure Dataset, Scientific Programming. Available at: https://www.hindawi.com/journals/sp/2022/3649406

[17] Online ICD9/ICD9CM codes. (n.d.) http://icd9.chrisendres.com/index.php?action=contents

[18] Pequenino, K. (2023, December 8). Investigadora portuguesa mostra como "dados desequilibrados" estão na raiz de falhas na IA. PÚBLICO. https://www.publico.pt/2023/12/08/ciencia/noticia/investigadora-portuguesa-mostra-dados-desequilibrados-estao-raiz-falhas-ia-2072806

[19] Random search for hyper-parameter optimization - journal of machine ... Available at: https://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

[20] Sanclemente, K., Kafer, J., Houlette, T., McDonnell,S. 2022. Lecture 5: Deep Learning: Logistic Regression. https://www.cs.rice.edu/~as143/COMP642_Spring22/Scribes/Lect5

[21] Santos, M. (2022). Research Problems in Data Quality Addressing Imbalanced and Missing Data [Doctoral Thesis]. Universidade de Coimbra.

[22] Senna, A. (2022). Predictive Modelling of Hospital Readmissions in Diabetic Patients Clusters [Dissertation]. Nova Information Management School.

[23] Singh, R. (2023, March 3). It's all about Outliers - Analytics Vidhya - Medium. Medium. https://medium.com/analytics-vidhya/its-all-about-outliers-cbe172aa1309

[24] sklearn.metrics.f1_score. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[25] Source of admission to an Inpatient facility — for newborn admit is type of delivery code | ResDAC. (n.d.). https://resdac.org/cms-data/variables/source-admission-inpatient-facility-newborn-admit-type-delivery-code

[26] Type of admission or visit codes - JE Part A - Noridian. (n.d.). JE Part A. https://med.noridianmedicare.com/web/jea/topics/claim-submission/type-of-admission-or-visit-codes

[27] What are Neural Networks? | IBM. (n.d.). https://www.ibm.com/topics/neural-networks

[28] Xue, Y. (2019). An Overview of Overfitting and its Solutions. Journal of Physics, 1168, 022022. https://doi.org/10.1088/1742-6596/1168/2/022022

10. Self-Study Techniques

10.1. Encoding

We use encoding to handle categorical variables, transforming them into numerical values and, therefore, making them compatible with machine learning algorithms. It's particularly useful when we are presented with a high number of categories.

10.1.1. Ordinal Encoding

Ordinal encoding is used to represent categorical variables with ordered relationships by assigning to each category a unique number. It is suitable for variables where the categories have a meaningful order to rank, such as customer satisfaction (e.g., lower, medium, high).

10.1.2. Target Encoding

Target encoding aims to replace each category with a measure of its impact on the target variable, using the mean as the representative measure.

10.1.3. Label Encoding

Label encoding assigns to each category of a categorical variable a unique integer label. It is used when the categories don't have a meaningful order, since it assumes no ordinal relationship between them.

10.2. Resampling Techniques

Resampling Techniques are methods used to deal with an imbalanced dataset, where one class significantly outnumbers the others. These techniques are usually implemented by adding or removing instances.

10.2.1. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is an oversampling technique that aims to address imbalance by creating synthetic examples for a class. It starts by identifying instances from the minority class, then it identifies their k-nearest neighbors and generates the synthetic examples by blending the features of both of them. These instances are added to the minority class, increasing its representation and balancing the dataset.

10.2.2. Synthetic Minority Oversampling Technique for Nominal and Continuous Features (SMOTE-NC)

Since SMOTE is originally designed for numerical features, we have **SMOTE-NC** as an extension of this algorithm. It operates in the same way but is able to recognize which features are numerical and which are categorical and then generate synthetic examples for both.

For the numerical features, it applies the SMOTE whereas in the categorical variables, it works by giving the value occurring in the majority of the k-nearest neighbors.

10.2.3. Synthetic Minority Oversampling Technique with Edited Nearest Neighbors (SMOTE-ENN)

SMOTE-ENN combines the advantages of both oversampling (SMOTE) and undersampling (ENN) methods. Initially, it applies SMOTE, in the same way it was previously explained, and then it applies ENN, a method that evaluates the k nearest neighbors of each majority class instance. If any of these neighbors are misclassified, they are removed, resulting in a more balanced dataset.

10.2.4. Random Undersampling

Random Undersampling works by randomly selecting instances from the majority class and removing them from the dataset. It does this until it achieves a balanced class distribution.

10.2.5. Random Oversampling

Random Oversampling basically works in the same way as random undersampling but, in this case, it increases the size of the minority class by duplicating instances at random.

10.3. KBest Mutual Information Classification

KBest using Mutual Information is a feature selection method that combines the SelectKBest method with mutual information as the scoring metric.

SelectKBest selects the best k features based on the highest scores according to a statistical test, with k being the number of top features we want to retain. It aims to retain only the most informative features and discard less relevant ones.

Mutual Information Classification is a measure of dependency between variables, this means that it is used to quantify the amount of information we can obtain about one variable by observing another. Higher values mean higher dependency.

With that being so, **KBest Mutual Information Classification** focuses on choosing a specific number of features by calculating the mutual information between each feature and the target variable and selecting the one that has the highest value.

10.4. Random Search and Grid Search

Random Search and Grid Search are two methods for optimizing hyperparameters, the key difference between them is how they choose the hyperparameters values. Grid Search explores the values systematically, evaluating the model for every combination and testing every possible scenario. In contrast, Random Search randomly selects the hyperparameter values for testing, so not all the parameters are tested.

10.5. Complement Naïve Bayes

The Complement Naive Bayes method calculates the probability of an instance not belonging to each class and then selects the smallest value (since the lowest probability of not belonging implies the highest probability of belonging to that class). This differs from the traditional Naive Bayes, as it calculates the probability of an item belonging to all classes, instead of only a specific one. It is also more suitable for imbalanced data by focusing on the complement class.

11. Annexes

Annex 1 – Data Information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71236 entries, 0 to 71235
Data columns (total 31 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   encounter_id                          71236 non-null  int64
 1   country                               71236 non-null  object
 2   patient_id                            71236 non-null  int64
 3   race                                  67682 non-null  object
 4   gender                                71236 non-null  object
 5   age                                   67679 non-null  object
 6   weight                                71236 non-null  object
 7   payer_code                            71236 non-null  object
 8   outpatient_visits_in_previous_year    71236 non-null  int64
 9   emergency_visits_in_previous_year     71236 non-null  int64
 10  inpatient_visits_in_previous_year     71236 non-null  int64
 11  admission_type                        67530 non-null  object
 12  medical_specialty                     71236 non-null  object
 13  average_pulse_bpm                     71236 non-null  int64
 14  discharge_disposition                 68646 non-null  object
 15  admission_source                      66518 non-null  object
 16  length_of_stay_in_hospital            71236 non-null  int64
 17  number_lab_tests                      71236 non-null  int64
 18  non_lab_procedures                    71236 non-null  int64
 19  number_of_medications                 71236 non-null  int64
 20  primary_diagnosis                     71236 non-null  object
 21  secondary_diagnosis                   71236 non-null  object
 22  additional_diagnosis                  71236 non-null  object
 23  number_diagnoses                      71236 non-null  int64
 24  glucose_test_result                   3688 non-null   object
 25  a1c_test_result                       11916 non-null  object
 26  change_in_meds_during_hospitalization 71236 non-null  object
 27  prescribed_diabetes_meds              71236 non-null  object
 28  medication                            71236 non-null  object
 29  readmitted_binary                     71236 non-null  object
 30  readmitted_multiclass                 71236 non-null  object
dtypes: int64(11), object(20)
memory usage: 16.8+ MB
```

Annex 2 – Descriptive Statistics (numerical variables)

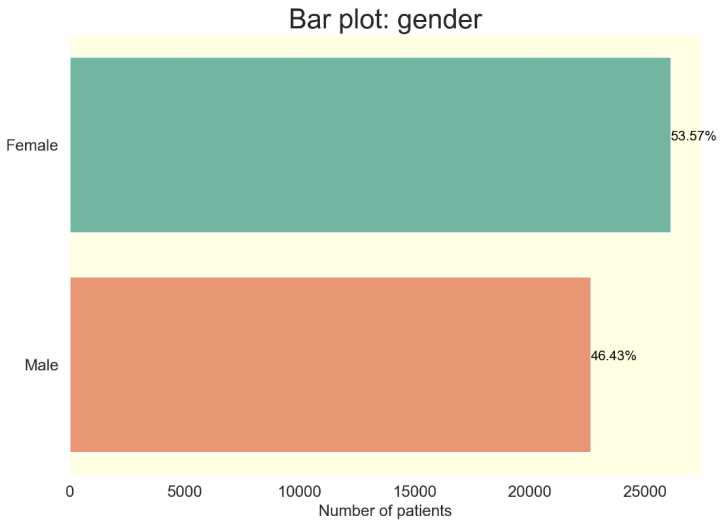
	count	mean	std	min	25%	50%	75%	max
encounter_id	49865.0	5.509153e+05	2.589922e+05	100011.0	326126.0	551493.0	774683.0	999979.0
patient_id	49865.0	5.429346e+07	3.881105e+07	135.0	23393988.0	45283518.0	87516477.0	189502619.0
outpatient_visits_in_previous_year	49865.0	3.650055e-01	1.263667e+00	0.0	0.0	0.0	0.0	42.0
emergency_visits_in_previous_year	49865.0	1.936027e-01	8.246386e-01	0.0	0.0	0.0	0.0	42.0
inpatient_visits_in_previous_year	49865.0	6.354557e-01	1.254075e+00	0.0	0.0	0.0	1.0	21.0
average_pulse_bpm	49865.0	9.948956e+01	2.311636e+01	60.0	79.0	100.0	120.0	139.0
length_of_stay_in_hospital	49865.0	4.397854e+00	2.992738e+00	1.0	2.0	4.0	6.0	14.0
number_lab_tests	49865.0	4.312797e+01	1.967118e+01	1.0	31.0	44.0	57.0	121.0
non_lab_procedures	49865.0	1.345593e+00	1.712298e+00	0.0	0.0	1.0	2.0	6.0
number_of_medications	49865.0	1.598442e+01	8.124069e+00	1.0	10.0	15.0	20.0	75.0
number_diagnoses	49865.0	7.417447e+00	1.940343e+00	1.0	6.0	8.0	9.0	16.0

Annex 3 – Descriptive Statistics (categorical variables)

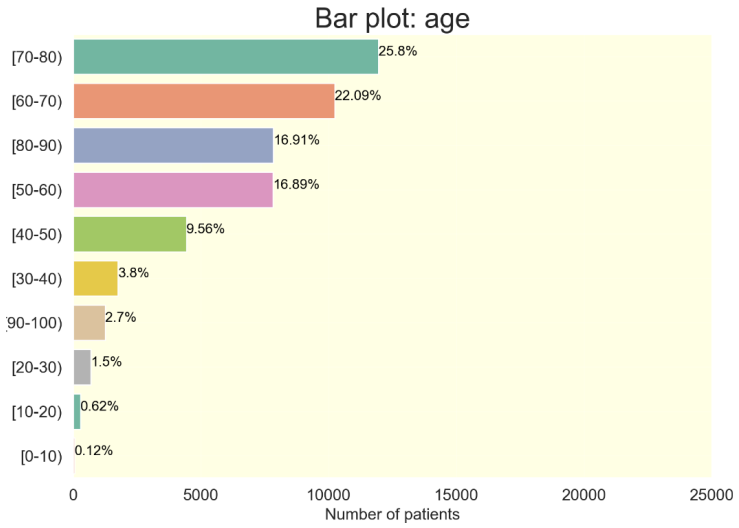
	count	unique	top	freq
country	49865	1	USA	49865
race	47400	6	Caucasian	35477
gender	49865	3	Female	26681
age	47385	10	[70-80)	12240
weight	49865	10	?	48304
payer_code	49865	17	?	19782
admission_type	47271	7	Emergency	26338
medical_specialty	49865	67	?	24385
discharge_disposition	48066	25	Discharged to home	29580
admission_source	46523	14	Emergency Room	28164
primary_diagnosis	49865	649	428	3397
secondary_diagnosis	49865	657	428	3275
additional_diagnosis	49865	706	250	5630
glucose_test_result	2559	3	Norm	1251
a1c_test_result	8343	3	>8	3997
change_in_meds_during_hospitalization	49865	2	No	26888
prescribed_diabetes_meds	49865	2	Yes	38448
medication	49865	273	['insulin']	15186
readmitted_binary	49865	2	No	44300

Annex 4 - Visualizations: Countplots

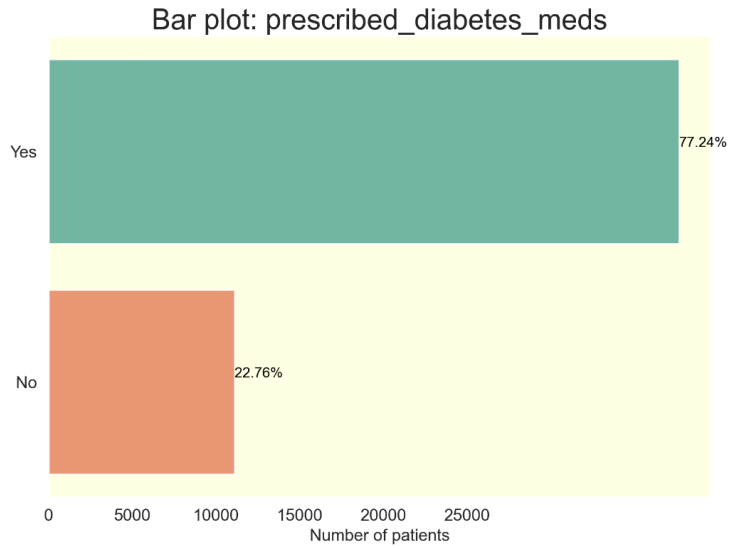
Annex 4.1. Gender Countplot



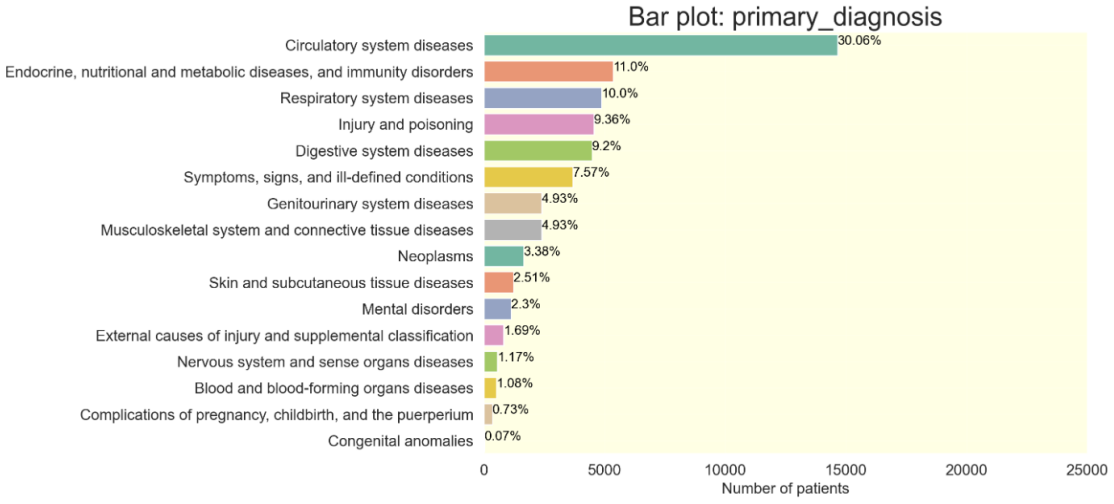
Annex 4.2. Age Countplot



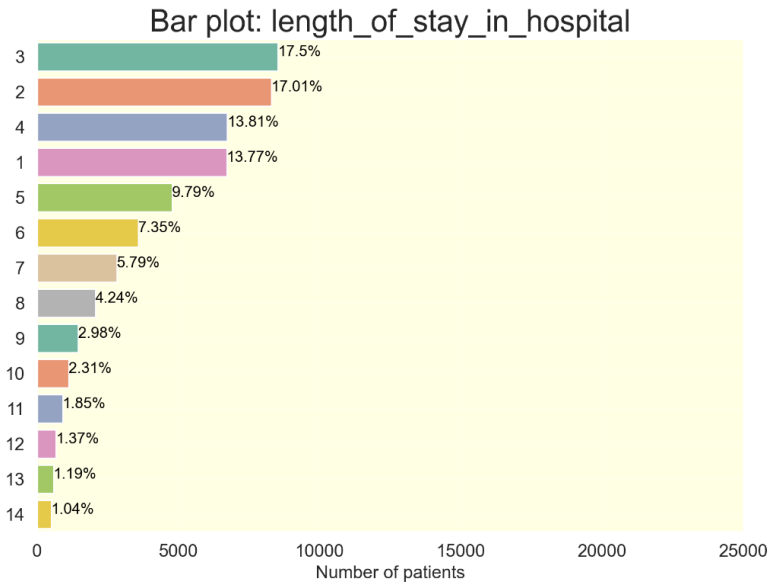
Annex 4.3. Prescribed Diabetes Medication Countplot



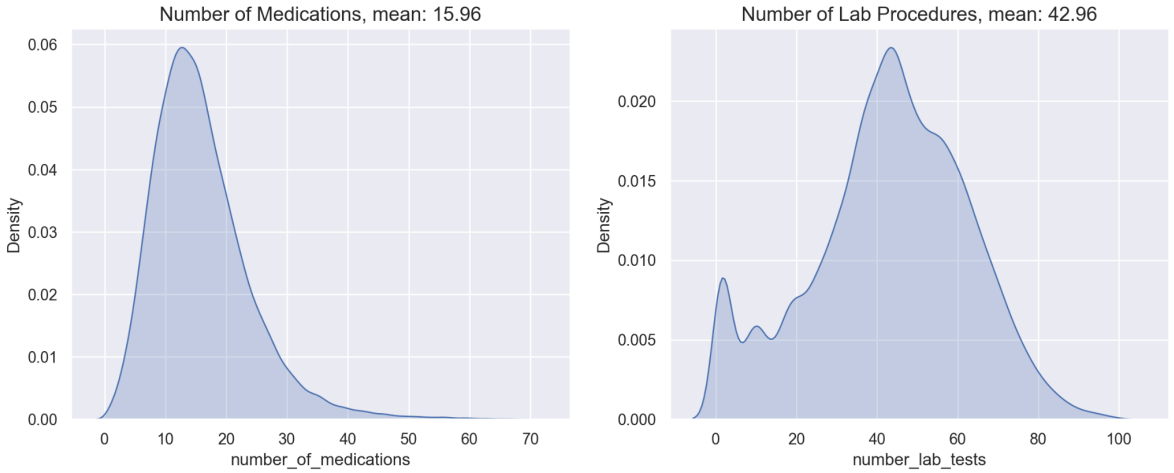
Annex 4.4. Primary Diagnosis Countplot



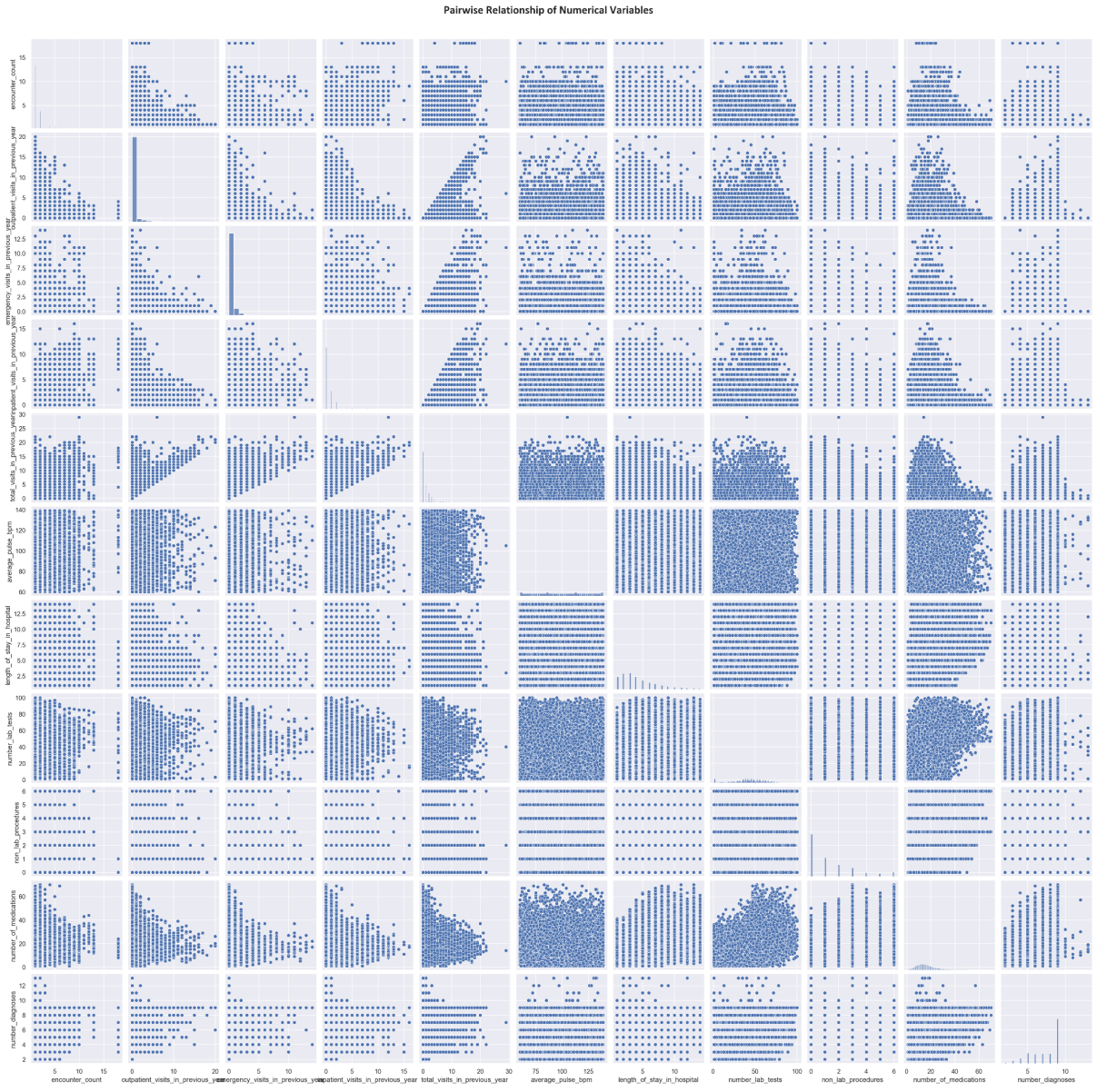
Annex 4.5. Length of Stay on Hospital Countplot



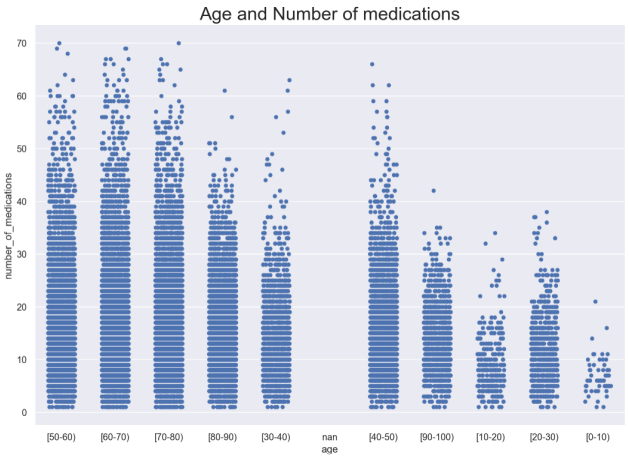
Annex 4.6. Number of Medication & Number of Lab Procedures Density Plot



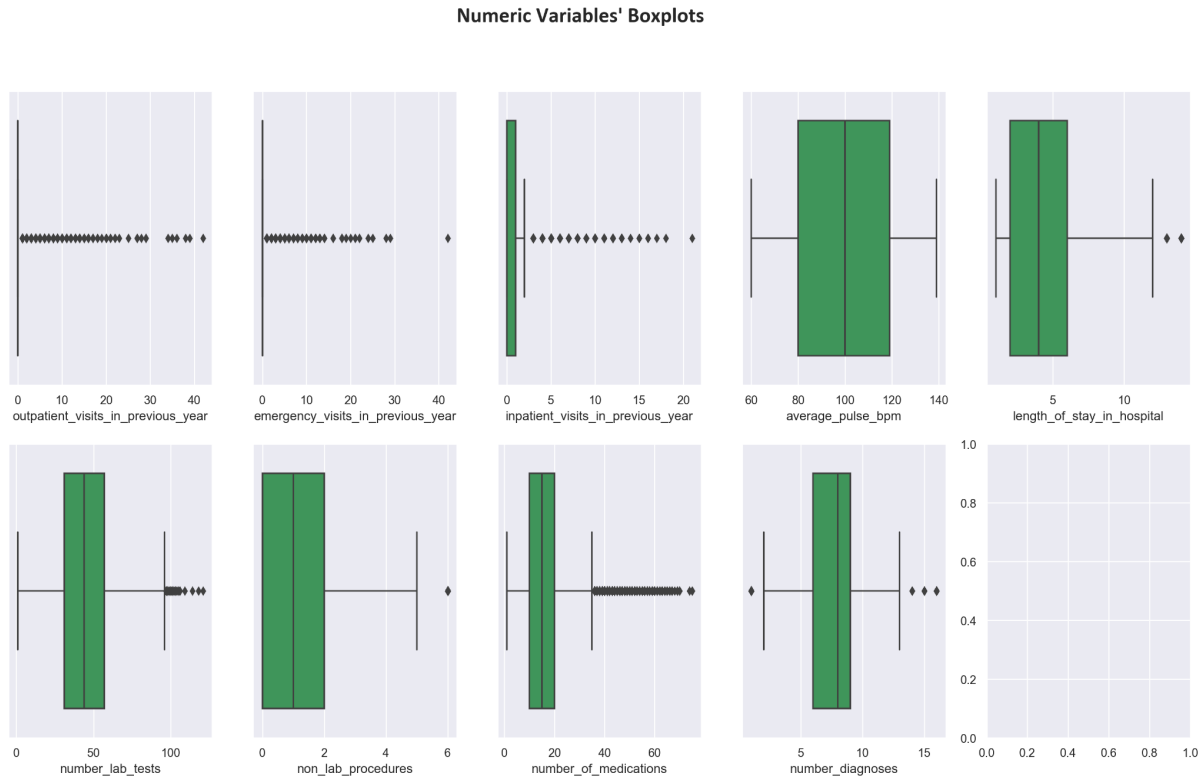
Annex 5 - Visualizations: Pairwise relationships of Numeric Variables



Annex 6 - Visualizations: Pairwise relationships between Age and Number of medications



Annex 7 – Outliers Detection



Annex 8 – Feature Engineering

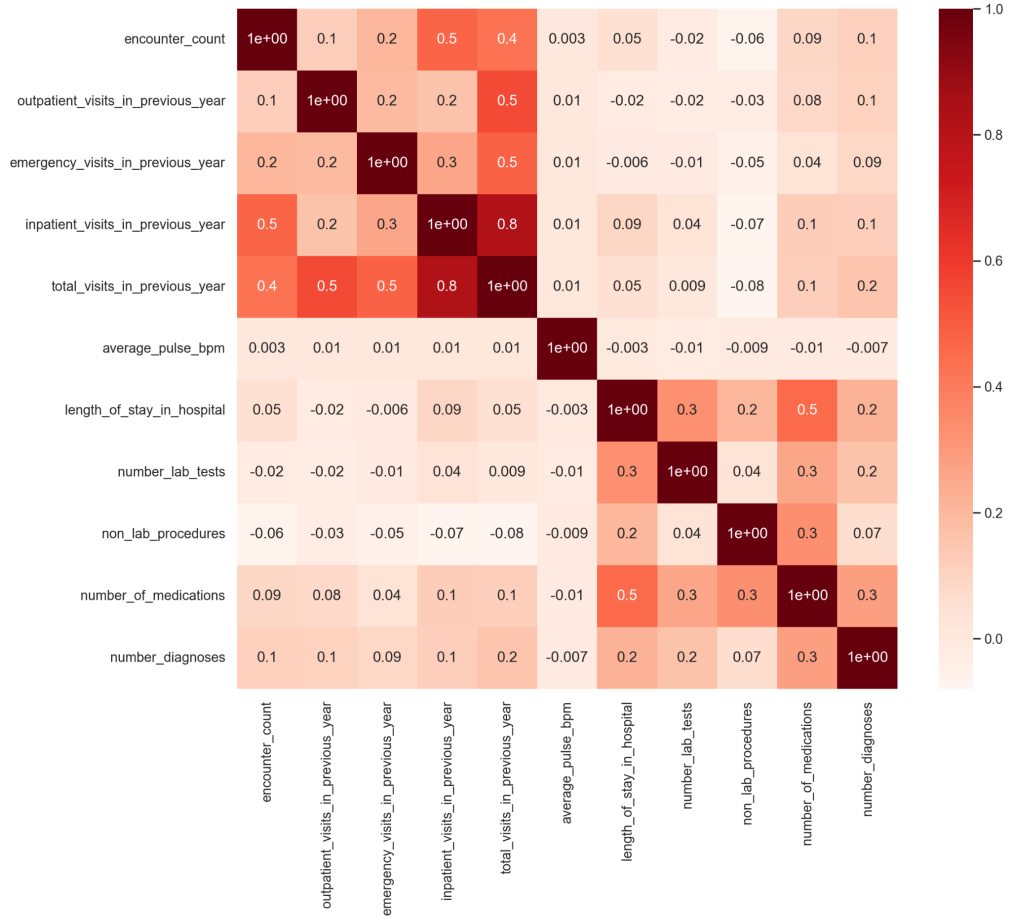
Variables	Explanation	Categories	Source
medical_specialty	Mapped into more general groups (28 categories), decreasing the number of categories in this feature (had a total of 67)	E.g.: 'GeneralSurgery', 'InternalMedicine' and 'Pediatrics'	Association of American Medical Colleges. (n.d.)
caucasian	Transforming *race,* it returns 1 if the patient is caucasian and 0 otherwise	1 if the patient is caucasian and 0 otherwise	_
primary_diagnosis, secondary_diagnosis and *additional_diagnosis*	The values should be a 3 digit code. Considered those numbers with only 1 or 2 digits as if they have zeros before (leading zeros are typically ignored)	E.g.: From code 1 to code 139, it returns the 'Infectious and parasitic diseases' category, from 140 to 239, it returns 'Neoplasms', and so on	ICD9/ICD9CM Codes
age	Using *age* by grouping into less categories (5 categories), it had a total of 10	'Child', 'Teen', 'Young Adult', 'Adult' and 'Elderly'	_
total_visits_in_previous_year (new variable)	Sum of *inpatient_visits_in_previous_year, outpatient_visits_in_previous_year and emergency_visits_in_previous_year*	E.g.: 5, if *outpatient_visits_in_previous_year = 1, emergency_visits_in_previous_year = 1 and inpatient_visits_in_previous_year = 3*	–
has_insurance (new variable)	Using the variable *payer_code* (with many categories, 18 in total) created a binary feature	1 if the individual has any type of insurance and 0 otherwise	–
type_II_diabetes (new variable)	Transformed the feature *medication* (271 categories) into a binary one. Type II diabetes' treatment is insulin combined with other medications, one of the other medications or combination.	1 if a patient has been prescribed a medication for type II diabetes or 0, which is considered 'Unknown' if it is ambiguous - values are '[]' or 'insulin' (can be prescribed to both type I or type II diabetes patients in one encounter)	–

Annex 9 – Binary Classification: Model Performance in different datasets

Dataset	Mode Imputation + Random Undersampling	Mode Imputation + SMOTE-NC	Mode Imputation + SMOTE-ENN	Mode Imputation + Random Oversampling and Random Undersampling
Gaussian Naïve Bayes	Train: 0.487	Train: 0.390	Train: 0.741	Train: 0.501
	Validation: 0.270	Validation: 0.232	Validation: 0.268	Validation: 0.274
	Difference: 0.216	Difference: 0.157	Difference: 0.472	Difference: 0.226
Complement Naïve Bayes	Train: 0.280	Train: 0.282	Train: 0.292	Train: 0.294
	Validation: 0.277	Validation: 0.272	Validation: 0.277	Validation: 0.281
	Difference: 0.003	Difference: 0.009	Difference: 0.015	Difference: 0.013
Logistic Regression	Train: 0.312	Train: 0.303	Train: 0.313	Train: 0.317
	Validation: 0.287	Validation: 0.271	Validation: 0.287	Validation: 0.290
	Difference: 0.024	Difference: 0.032	Difference: 0.026	Difference: 0.027
Decision Trees	Train: 0.313	Train: 0.311	Train: 0.313	Train: 0.270
	Validation: 0.309	Validation: 0.290	Validation: 0.294	Validation: 0.263
	Difference: 0.004	Difference: 0.021	Difference: 0.019	Difference: 0.006
K-Nearest Neighbors	Train: 0.599	Train: 0.847	Train: 0.881	Train: 0.756
	Validation: 0.251	Validation: 0.220	Validation: 0.227	Validation: 0.234
	Difference: 0.348	Difference: 0.626	Difference: 0.654	Difference: 0.522
Neural Networks	Train: 0.620	Train: 0.592	Train: 0.813	Train: 0.628
	Validation: 0.286	Validation: 0.261	Validation: 0.238	Validation: 0.284
	Difference: 0.334	Difference: 0.330	Difference: 0.574	Difference: 0.343
Support Vector Machines	Train: 0.332	Train: 0.294	Train: 0.328	Train: 0.327
	Validation 0.303	Validation 0.250	Validation 0.297	Validation 0.291
	Difference: 0.028	Difference: 0.044	Difference: 0.031	Difference: 0.036
Random Forest	Train: 0.314	Train: 0.314	Train: 0.311	Train: 0.310
	Validation 0.302	Validation 0.299	Validation 0.298	Validation 0.295
	Difference: 0.011	Difference: 0.014	Difference: 0.012	Difference: 0.015

Dataset	KNN Imputation + Random Undersampling	KNN Imputation + SMOTE-NC	KNN Imputation + SMOTE-ENN	KNN Imputation + Random Oversampling and Random Undersampling
Gaussian Naïve Bayes	Train: 0.480	Train: 0.542	Train: 0.699	Train: 0.516
	Validation: 0.267	Validation: 0.256	Validation: 0.266	Validation: 0.276
	Difference: 0.213	Difference: 0.285	Difference: 0.432	Difference: 0.240
Complement Naïve Bayes	Train: 0.276	Train: 0.282	Train: 0.283	Train: 0.295
	Validation: 0.267	Validation: 0.269	Validation: 0.270	Validation: 0.281
	Difference: 0.008	Difference: 0.013	Difference: 0.012	Difference: 0.014
Logistic Regression	Train: 0.313	Train: 0.307	Train: 0.305	Train: 0.317
	Validation: 0.286	Validation: 0.275	Validation: 0.272	Validation: 0.289
	Difference: 0.027	Difference: 0.031	Difference: 0.032	Difference: 0.028
Decision Trees	Train: 0.314	Train: 0.309	Train: 0.315	Train: 0.314
	Validation: 0.289	Validation: 0.290	Validation: 0.286	Validation: 0.289
	Difference: 0.024	Difference: 0.018	Difference: 0.028	Difference: 0.024
K-Nearest Neighbors	Train: 0.620	Train: 0.795	Train: 0.878	Train: 0.748
	Validation: 0.257	Validation: 0.239	Validation: 0.226	Validation: 0.228
	Difference: 0.362	Difference: 0.555	Difference: 0.651	Difference: 0.519
Neural Networks	Train: 0.591	Train: 0.646	Train: 0.814	Train: 0.552
	Validation: 0.242	Validation: 0.229	Validation: 0.240	Validation: 0.214
	Difference: 0.349	Difference: 0.417	Difference: 0.574	Difference: 0.338
Support Vector Machines	Train: 0.327	Train: 0.287	Train: 0.310	Train: 0.322
	Validation: 0.295	Validation 0.242	Validation 0.268	Validation 0.291
	Difference: 0.032	Difference: 0.045	Difference: 0.042	Difference: 0.031
Random Forest	Train: 0.315	Train: 0.320	Train: 0.203	Train: 0.319
	Validation: 0.302	Validation: 0.301	Validation 0.198	Validation: 0.311
	Difference: 0.013	Difference: 0.019	Difference: 0.005	Difference: 0.008

Annex 10 – Spearman Correlation Matrix



Annex 11 – Binary Classification: Feature Selection

	Univariate Variables	Spearman Correlation	ChiSquare	RFE	LASSO	Random Forest	Kbest (Mutual Info)	%Keep	Final Decision
age	-	-	Keep	Discard	Keep	Keep	Discard	0.6	Keep
gender	-	-	Discard	Discard	Keep	Discard	Discard	0.2	Discard
payer_code	-	-	Keep	Discard	Keep	Discard	Discard	0.4	Discard
outpatient_visits_in_previous_year	Keep	Keep	-	Discard	Discard	Discard	Discard	0.333	Discard
emergency_visits_in_previous_year	Keep	Keep	-	Discard	Keep	Discard	Keep	0.667	Keep
inpatient_visits_in_previous_year	Keep	Keep	-	Keep	Keep	Keep	Keep	1	Keep
admission_type	-	-	Discard	Discard	Keep	Discard	Keep	0.4	Discard
medical_specialty	-	-	Keep	Discard	Keep	Discard	Keep	0.6	Keep
average_pulse_bpm	Keep	Discard	-	Discard	Keep	Keep	Discard	0.5	Discard
discharge_disposition	-	-	Keep	Keep	Keep	Keep	Keep	1	Keep
admission_source	-	-	Keep	Discard	Keep	Discard	Keep	0.6	Keep
length_of_stay_in_hospital	Keep	Keep	-	Discard	Keep	Keep	Discard	0.667	Keep
number_lab_tests	Keep	Keep	-	Discard	Keep	Keep	Discard	0.667	Keep
non_lab_procedures	Keep	Keep	-	Discard	Keep	Discard	Discard	0.5	Discard
number_of_medications	Keep	Keep	-	Discard	Keep	Keep	Discard	0.667	Keep
primary_diagnosis	-	-	Keep	Discard	Keep	Keep	Discard	0.6	Keep
secondary_diagnosis	-	-	Keep	Discard	Keep	Keep	Discard	0.6	Keep
additional_diagnosis	-	-	Keep	Discard	Keep	Keep	Discard	0.6	Keep
number_diagnoses	Keep	Keep	-	Discard	Keep	Discard	Discard	0.5	Discard
glucose_test_result	-	-	Discard	Discard	Keep	Discard	Keep	0.4	Discard
a1c_test_result	-	-	Keep	Discard	Keep	Discard	Discard	0.4	Discard
change_in_meds_during_hospitalization	-	-	Keep	Discard	Keep	Discard	Discard	0.4	Discard
prescribed_diabetes_meds	-	-	Keep	Discard	Keep	Discard	Keep	0.6	Keep
medication	-	-	Keep	Discard	Keep	Keep	Discard	0.6	Keep
has_insurance	-	-	Discard	Discard	Keep	Discard	Discard	0.2	Discard
age_groups		-	Keep	Discard	Keep	Discard	Discard	0.4	Discard
total_visits_in_previous_year	Keep	Discard	-	Discard	Keep	Keep	Keep	0.667	Keep
type_II_diabetes	-	-	Keep	Discard	Keep	Discard	Discard	0.4	Discard
caucasian	-	-	Discard	Discard	Keep	Discard	Discard	0.2	Discard
encounter_count	Keep	Keep	-	Keep	Keep	Keep	Keep	1	Keep

Annex 12 – Multiclass Classification: LASSO results

LASSO	
encounter_count	-2.788852
inpatient_visits_in_previous_year	-0.422740
total_visits_in_previous_year	-0.237096
number_diagnoses	-0.200253
age_groups	-0.162913
emergency_visits_in_previous_year	-0.159417
number_lab_tests	-0.097929
number_of_medications	-0.043535
caucasian	-0.020445
prescribed_diabetes_meds	-0.017629
type_II_diabetes	-0.002178
average_pulse_bpm	-0.001902
outpatient_visits_in_previous_year	0.000000
admission_source	0.000020
glucose_test_result	0.000327
change_in_meds_during_hospitalization	0.001271
gender	0.007392
length_of_stay_in_hospital	0.011092
non_lab_procedures	0.027439
a1c_test_result	0.041473
admission_type	0.071305
has_insurance	0.108358
primary_diagnosis	0.122544
additional_diagnosis	0.149451
secondary_diagnosis	0.234349
medical_specialty	0.246284
payer_code	0.289275
medication	0.349007
discharge_disposition	1.081155

Annex 13 – Multiclass Classification: Feature Selection

	Univariate Variables	Spearman Correlation	ChiSquare	KBest (Mutual Info)	RFE	LASSO	Random Forest	%Keep	Final Decision
gender	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
payer_code	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
outpatient_visits_in_previous_year	Keep	Keep	-	Discard	Keep	Discard	Discard	0,5	Discard
emergency_visits_in_previous_year	Keep	Keep	-	Keep	Keep	Keep	Discard	0,833333333	Keep
inpatient_visits_in_previous_year	Keep	Keep	-	Keep	Keep	Keep	Discard	0,833333333	Keep
admission_type	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
medical_specialty	-	-	Keep	Keep	Keep	Keep	Discard	0,8	Keep
average_pulse_bpm	Keep	Discard	-	Discard	Keep	Keep	Keep	0,666666667	Keep
discharge_disposition	-	-	Keep	Keep	Keep	Keep	Keep	1	Keep
admission_source	-	-	Keep	Keep	Keep	Discard	Discard	0,6	Keep
length_of_stay_in_hospital	Keep	Keep	-	Discard	Keep	Keep	Keep	0,833333333	Keep
number_lab_tests	Keep	Keep	-	Discard	Keep	Keep	Keep	0,833333333	Keep
non_lab_procedures	Keep	Keep	-	Discard	Keep	Keep	Keep	0,833333333	Keep
number_of_medications	Keep	Keep	-	Discard	Keep	Keep	Keep	0,833333333	Keep
primary_diagnosis	Keep	-	Keep	Keep	Keep	Keep	Keep	1	Keep
secondary_diagnosis	-	-	Keep	Discard	Keep	Keep	Keep	0,8	Keep
additional_diagnosis	-	-	Keep	Discard	Keep	Keep	Keep	0,8	Keep
number_diagnoses	-	Keep	-	Keep	Keep	Keep	Keep	1	Keep
glucose_test_result	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
a1c_test_result	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
change_in_meds_during_hospitalization	-	-	Keep	Discard	Discard	Discard	Discard	0,2	Discard
prescribed_diabetes_meds	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
medication	-	-	Keep	Discard	Keep	Keep	Keep	0,8	Keep
has_insurance	-	-	Keep	Keep	Keep	Keep	Discard	0,8	Keep
age_groups	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
total_visits_in_previous_year	Keep	Discard	-	Keep	Keep	Keep	Discard	0,666666667	Keep
type_II_diabetes	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
caucasian	-	-	Keep	Discard	Keep	Keep	Discard	0,6	Keep
encounter_count	Keep	Keep	-	Keep	Keep	Keep	Keep	1	Keep

Annex 14 – Multiclass Classification: Model Performance Table with Imbalance Handling Methods

Dataset	KNN Imputation + Random Undersampling	KNN Imputation + SMOTE-NC	KNN Imputation + Random Oversampling and Random Undersampling
Logistic Regression	Train: 0.489	Train: 0.469	Train: 0.490
	Validation: 0.524	Validation: 0.509	Validation: 0.527
	Difference: -0.034	Difference: -0.040	Difference: -0.038
Multinomial Naive Bayes	Train: 0.523	Train: 0.449	Train: 0.458
	Validation: 0.545	Validation: 0.509	Validation: 0.508
	Difference: -0.023	Difference: -0.060	Difference: -0.050
Decision Trees	Train: 0.459	Train: 0.320	Train: 0.466
	Validation: 0.508	Validation: 0.247	Validation: 0.512
	Difference: -0.049	Difference: 0.072	Difference: -0.047
K-Nearest Neighbors	Train: 0.575	Train: 0.575	Train: 0.575
	Validation: 0.502	Validation: 0.490	Validation: 0.502
	Difference: 0.073	Difference: 0.085	Difference: 0.073
Neural Networks	Train: 0.523	Train: 0.517	Train: 0.546
	Validation: 0.545	Validation: 0.515	Validation: 0.564
	Difference: -0.023	Difference: 0.002	Difference: -0.018
Random Forest	Train: 0.488	Train: 0.554	Train: 0.498
	Validation: 0.512	Validation: 0.539	Validation: 0.526
	Difference: -0.024	Difference: 0.015	Difference: -0.028
Support Vector Machines	Train: 0.583	Train: 0.571	Train: 0.583
	Validation 0.525	Validation 0.518	Validation 0.525
	Difference: 0.057	Difference: 0.053	Difference: 0.056

Annex 15 – ROC Curve Analysis (Binary Classification)