NOVA IMS Information Management School

Machine Learning Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

READY TO BE DISCHARGED: EXAMINING HOSPITAL READMISSIONS

Group 32

Andriani Kakoulli, number: 20230484

Beatriz Xavier, number: 20230987

Diogo Barros, number: 20230555

Filipe Pereira, number: 20230445

Pedro Luís, number: 20230797

December 2023

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

INDEX

Abstract

Hospital readmissions are of utmost importance to assess healthcare quality, especially regarding chronic illnesses such as diabetes. In light of this, it is logical to investigate the challenge presented by readmissions, as it can have a great impact not only on patient care but also on healthcare costs.

The objective of this project is to leverage machine learning algorithms for the development of predictive models capable of accurately forecasting hospital readmissions for diabetic patients. To this end, our work was twofold: firstly, we implemented a binary classification model that predicts whether a patient will be readmitted within 30 days of discharge and, secondly, we developed a multiclass classification model that categorizes the readmission timeframe as *"NO"*, *"<30days"* and *">30days"*. The binary classifier seeks to aid healthcare providers in implementing specific preventative measures, whilst the multiclass classifier provides detailed insights into varying levels of patient risk, enabling more customized post-discharge care.

In order to achieve our goal, we used a comprehensive dataset with 71236 records in total, comprising 53985 unique patients with eclectic demographic details, medical history and treatment records. This data was thoroughly explored, preprocessed and fine-tuned, and different models were tested for the purpose of ensuring robustness and high predictive accuracy.

The results of this project revealed to be promising to decrease readmission rates and improve patient care. The models showed potential in recognizing patients that are at risk for readmission, which is crucial to improve resource allocation and financial decisions. On close analysis, it is evident that predictive analytics are critical in healthcare, and that future research should be conducted to further enhance and broaden the applicability of these predictive models.

1. Introduction

This project aims to develop machine learning models for both binary and multiclass classification problems. We know that hospital readmissions are sometimes a hard challenge to control correctly both as an indicator of care quality and as a factor of increasing costs. The main goal of this project is to accurately predict if a patient will be readmitted to the hospital within 30 days of being discharged, for the binary classification, and to predict the timeframe of a patient's readmission between *"NO"*, *"<30days"*, *">30days"*, for the multiclass classification. The purpose of these models is to aid healthcare providers in making decisions on preventive measures and reduce healthcare costs.

In order to prepare for this project, we sought to review some existing literature on the subject [Paper1, Paper2, Paper3]. This gave us insights not only on the dataset we were working with, but also on some practical approaches to build our models. The three studies underscore the importance of detailed data pre-processing and the selection of appropriate modeling techniques. Taking these approaches and their respective predictive factors into consideration was helpful in guiding us towards achieving better accuracy and insightful results. Depending on the specific features on our dataset, we drew the conclusion that we could also expect to find certain features, like *a1c_test_results* and *discharge_disposition*, as significant predictors in our model.

In the following sections, we will delve into the methodologies employed, the nature of the dataset and preprocessing steps, the development and results of the classification models, and a comprehensive discussion of our findings. This report aims to provide a thorough description of the project, from conceptualization to execution, and its potential impact on healthcare management for diabetic patients.

2. Data Exploration

For this project, we utilized a detailed dataset that included both encounter-level data, such as a unique encounter ID (*encounter_id*), and the patient's personal information. The latter incorporates patient demographics (e.g. *age*, *race*, *gender*), medical history (e.g. *length_of_stay_in_hospital*, admission_*source*) and clinical information (e.g. *primary_diagnosis*, *medication*, *glucose_test_result*). This extensive dataset offered a detailed perspective on the patients, which is crucial to create precise predictive models.

We started by exploring the dataset to get insights about features and data types, as well as understand what we possibly had to take into consideration to get the best possible results. There were several noticeable aspects of interest in our dataset. For example, *country* only had one value, so we were able to drop it right away because it wouldn't have any influence on our predictions. The feature *patient_id* had duplicated values, which were useful later for feature engineering. We also detected some uncommon values, such as 'Unknown/Invalid', 'Not Available', 'Not Mapped' and '?', which were replaced with missing values as it sounded more logical. We also observed some trailing blank spaces in *admission_source* that were removed. Since the feature *age* represented the patient's age in brackets, we decided to transform it into numerical by replacing each bracket with the midpoint of each category. Finally, *encounter_id* was set as the index because it was the unique identifier.

To get a sense of the target variable for binary classification, we plotted it and noticed that 88.8% are negative results (Figure 1), which means we are dealing with an imbalanced dataset. This will be dealt with later in our modeling. The target variable was encoded as numeric at this stage.

3. Preprocessing

In the first stage of data preprocessing, we cleaned and organized the data extensively. This work involved the handling of missing values, removing duplicates and transforming categorical data to match the format of our predictive algorithms. More precisely, we applied imputation methods to treat missing values and encoding methods for features such as *age*, *race*, *gender* and *medical codes*.

3.1 Missing Values

In order to better understand how we should handle the missing values, we created a data frame showing the percentage of missing values in each feature (Figure 2). This revealed some interesting results in *weight*, *glucose_test_result* and *a1c_test_result*, which had 96.85%, 94.82% and 83.27% of missing values, respectively.

After this, we began replacing NA values in each feature. Firstly, the missing values of the feature *race* were filled in with 'Unknown' rather than being aggregated in the existing value of 'Other', due to the possible presence of the existing categories in the missing data. The variable *gender* only had 3 missing values, so on a first approach one might argue we could drop these rows, because we wouldn't be losing a lot of information. However, we ended up filling these missing values with the mode for the sake of not having problems later when dealing with the test set. The feature *age* required a visualization of its distribution (Figure 3) to understand what imputation method was the most suitable. As we can observe, the data does not appear to be normally distributed, so KNN or an imputation of the median are both suitable approaches. We chose to use the median. *Weight* was dropped because it was 97% missing data and that, even though it can be an important metric in this dataset, the absence of accompanying factors such as the patient's height make the variable of weight less informative and thus less important for the predictions. As mentioned above, *glucose_test_result* and *a1c_test_result* had a high percentage of missing data. However, upon doing some further research on the subject, we considered that not only these are essential tests performed to diagnose diabetes, but also that these missing values only meant that the test was not taken. Understanding this, we decided to replace them with 'Not Taken'. For the rest of the features, we just replaced all the missing values with 'Unknown', as we figured it would be the best approach to not unbalance the data.

3.2 Reducing cardinality of categorical features

Perceiving what is happening in each feature is just as important as figuring out and handling problems with the data. For instance, reducing the cardinality of the categorical features is crucial to optimize the model's predictive performance and to increase interpretability. Reducing dimensionality also helps enhance computational efficiency and decreases the risk of overfitting.

Before anything else, we printed the number of categories in each feature and noticed that there were a few (e.g. *primary_diagnosis*, *additional_diagnosis*, ...) with around 700 different answers. With the aim of getting a grasp of the distribution of the categories in each feature, we plotted some count plots and pie charts as it will be mentioned afterwards. Some of the plots ended up giving us ideas on how to manage the number of categories too.

For the feature *payer_code*, we also chose to plot the relationship between its value counts and the probability of readmission (Figure 4). Nonetheless, no patterns were observed in the data, leading us to the decision of handling this attribute through feature engineering. We assumed that the 'Unknown' values filled in before meant that the patient doesn't have insurance.

Following this, we observed that the feature *admission_type* had two categories with a very low prevalence in this dataset (Figure 5). We opted to join '*Trauma Center*' with '*Emergency*' and '*Newborn*' with '*Urgent*', because these categories share similar characteristics in a medical context. Despite the high number of categories in *medical_specialty*, we decided to maintain the existing structure, as grouping them based on their similarities led to a great loss of detail. For *discharge_disposition*, we recognized that any category with the word 'expired' meant that the patient had passed away, which naturally meant that he would not be readmitted, therefore, we grouped those categories as just one, named '*Expired*'. We used the same approach for *admission_source*, grouping them by medical context. We chose to keep '*Emergency Room*', grouped the ones related to referrals as '*Referral*' and the ones related to transfers as 'Transfer', and finally joined the remaining ones in a category named '*Other*'.

Afterwards, we analyzed *primary_diagnosis*, *secondary_diagnosis* and *additional_diagnosis*. Before doing any kind of visualization, we had to reduce the number of categories, as these 3 variables had a high cardinality. For this, we used the mapping based on the ICD9 codes (Figure 6). This reduction in dimension seemed to be sufficient at this stage. The feature *medication* had the same problem as the former, but in this case, we decided to create binary variables for each unique medication (and drop the initial attribute of *medication*), representing if that medication was or wasn't prescribed to the patient. Since we observed a lot of columns where one of the classes had very low prevalence (Figure 7), we dropped all the columns of medication where this prevalence was lower than 1%.

Later, we mapped *glucose_test_result* and transformed it into a numerical feature because we can interpret the data as having an "order". The *a1c_test_result* proved to have a more complex analysis: in the cat plot (Figure 8), we encountered some strange information. If the A1C level is higher than 8, it is considered suboptimal and associated with a higher risk of diabetes-related complications, therefore we should be able to assume that the patients in this range would be more likely to be readmitted, and the same should be true if the A1C level is higher than 7, but with a lower risk of diabetes-related complications. However, as we can see by the cat plot, the patients within these ranges did not show a higher probability of being readmitted, on the contrary: they had about the same probability of being readmitted as the patients in the 'Norm'. Moreover, the patients who had not taken the test were a little more likely to be readmitted. Based on this information, we put the original variable in a numerical format as we did for *glucose_test_result* and later, we addressed this variable through feature engineering.

3.3 Exploring correlation between numerical features

Initially, we checked for the correlation between all the pairs of numerical features (Figure 9). The correlation heatmap showed some correlation between certain variables, with the highest correlation of 0.46 noted between '*number_of_medications*' and '*length_of_stay_in_hospital*'. This is logical, since in medical facilities people are usually provided with enhanced care, which can consist of more medication. However, it didn't seem like something to be concerned about.

3.4 Outliers in numerical features

The final step for preprocessing the data was to check for outliers. We plotted both histograms and boxplots for the numerical features (Figure 10, Figure 11) and noticed that we got some outliers for almost all the variables, and that some variables exhibit high skewness. To limit the impact of the outliers on our data, we decided to apply winsorizing. This is a statistical technique that moderates the influence of outliers on the mean and variance by setting extreme values to be within a certain range, therefore creating more robust estimators. As we can see (Figure 12), we managed to reduce quite significantly the number of outliers on this dataset and decided to keep the ones that remained.

4. Feature Engineering

During the data exploration and preprocessing steps, we gained a deep understanding of our dataset. We opted to move on to feature engineering with the goal of improving the model's performance, and since we had already modified and encoded some of the existing features, we now proceeded by focusing on feature creation.

One of our first actions was to merge the number of outpatient, emergency and inpatient visits the patient made to the hospital in previous years into a new feature named *service_utilization_in_previous_year*.

While managing the cardinality of categorical features, we reached several conclusions. One of them resulted in the creation of the variable *has_insurance*, that indicates whether the patient has insurance or not, since no discernible patterns emerged while observing the attribute of *payer_code*. Apart from the actions mentioned previously on the feature *medication*, we created the binary column of *prescribed_meds,* stating whether a medication has been prescribed to the patient, and the numerical column of *num_prescribed_meds,* indicating the number of medications that were prescribed to the patient. As for the variables of *glucose_test_result* and *a1c_test_result*, we created two binary variables named *glucose_test_taken* and *a1c_test_taken*, that represent if the patient has taken each of the tests, respectively, as there were obvious patterns of readmission when the patient had taken these tests.

We observed that *patiend_id* had duplicated values, so we decided to take advantage of this observation by creating some relevant variables. The new binary feature of *patient_dup* states whether the patient is duplicated and the numerical feature of *num_patient_dup* gives the number of times the patient is duplicated.

Furthermore, based on the previously mentioned observation and concerning the multiclass classification, three more features were created: *num_readmissions*, which indicates the number of patient's readmissions, *num_readmissions_in30*, which states the number of readmissions within 30 days of being discharged, and *num_readmissions_over30*, which states the number of readmissions after 30 days of being discharged.

4.1 Additional Preprocessing for Multiclass

As this section of our work was carried out after the completion of binary classification, we didn't consider it necessary to restart the whole preprocessing and data cleaning steps. This had to do with a variety of different factors. Firstly, the default features were the same for both classifications, and data preprocessing steps such as handling missing values and encoding categorical features can be used across multiple types of classification. Secondly, it is of key importance to consider efficiency and computational resources when working with machine learning, and restarting the whole process would probably be redundant. Finally, keeping the data as consistent as possible between classifications makes it easier to understand and compare model performances.

It was expected that not all the features would have the same importance in binary and multiclass classification. With this in mind, we started by getting closer to what was happening now among the features and the new target variable (*"NO"*, *"<30days"* and *">30days"*). Firstly, we plotted the probability of each value in the target variable for each group in each categorical feature (e.g. Figure 13). It came to our attention that a lot of features do not visually appear to be relevant for the prediction, since the distribution of the target variable remains roughly the same across different groups of said features. This may suggest that certain features don't have a strong predictive power, but in an effort to not take any rash decisions, we decided to keep analyzing.

As we saw previously, some of the numerical features had skewed distribution, so we decided to apply box-cox transformation for the ones with no zeros, which were *length_of_stay_in_hospital* and *number_of_medications* (Figure 14 and Figure 15).

4.2 Feature Selection for Binary Classification

When it comes to additional preprocessing, we tried to use other methods of feature selection, such as *RFECV*, *Boruta*, *Chi-squared* and *Lasso*. However, all these tests proved to not give us better results when it came to the F1 score, so we decided to keep all the default features for the predictions.

4.3 Feature Selection for Multiclass Classification

We proceeded by using the chi-squared test in the categorical features as a continuation of the analysis, and the feature *glyburide* proved not to be statistically significant because the p-value was 0.21. It was also important to know the level of correlation of each feature with the target variable. Considering that *average_pulse_bpm* had a correlation of -0.0042 and a poor visual significance, we decided to remove it.

5. Modeling

5.1 Model selection

We have now reached the point in our project where we need to choose the right model for our predictions (the following description was applied in both binary and multiclass classifications). To do this, we firstly needed to split the target variable from the rest of the features for both binary and multiclass problems. On a first approach, the variables *num_readmissions, num_readmissions_in30* and *num_readmissions_over30* were omitted with the aim of preventing data leakage. After this, we encoded the categorical variables with *LabelEncoder()*, which is a class used to transform categorical labels into numerical values. The numerical variables were also encoded with *PowerTransformer()* (which is a class that applies a power transformation to make the data more similar to a gaussian function).

5.1.1 Binary Classification

Before training the model, we decided to see some resampling techniques due to the dataset imbalance. We tried techniques like *SMOTE, RandomOverSampler, and RandomUnderSampler. Synthetic Minority Over-sampling Technique (SMOTE)* creates synthetic examples of the minority class. This might cause the model to overfit to these specific examples. *RandomOverSampler* is a method in machine learning used to address class imbalance by randomly duplicating instances from the minority classes until a more balanced distribution is achieved. Of these three methods, the *RandomUnderSampler* was the one that gave the best results. This method is a resampling technique in machine learning designed to alleviate class imbalance in datasets. It randomly removes instances from the majority class or classes until a more balanced class distribution is obtained. After choosing the resampling technique we also tried different sampling strategies for the *RandomUnderSampler* and 0.625 was the best one.

Following this, we created a cross-validation function to implement in our models, using the Stratified K-Fold Cross Validation. This resampling technique involves the splitting of the training set into *k* smaller sets, whilst the test set is still held out. The model is trained using *k-1* folds of the data and validated using the remaining part of the data. We decided to set the number of folds, *k*, to 5, which is a value that is usually enough to ensure consistent results, whilst not being computationally expensive. Then, still using our cross-validation function, a list of models was trained and the F1 score was evaluated. This list included *LogisticRegression*, *RandomForestClassifier*, *GradientBoostingClassifier*, *SVC*, *GaussianNB*, *DecisionTreeClassifier*, *AdaBoostClassifier*, *MLPClassifier* and *HistGradientBoostingClassifier*. The reason why we chose to evaluate each model by its F1 score was because this metric takes an equal relative contribution of precision and recall. To be more time efficient, we chose to limit our list of models for the final submission to the ones that gave the best F1 scores and ended up testing the models of *RandomForestClassifier*, *GradientBoostingClassifier*, *AdaBoostClassifier* and *HistGradientBoostingClassifier*.

Among the F1 scores of the evaluated classifiers, the ensemble method of the Gradient Boosting Classifier emerged as the top performing model with a score of 0.36, without using the three final variables created: *num_readmissions*, *num_readmissions_in30* and *num_readmissions_over30*. Unfortunately, after performing the same steps for our models with these features, we concluded that they introduced data leakage, since they outperformed the previous training and validation sets but got considerably worse results in the test sets.

5.1.2 Multiclass Classification

Following the procedures described above and after dividing the data with *train_test_split*, we applied RFECV using a Stratified K-Fold Cross Validation where k=5, in order to improve efficiency and effectiveness by eliminating irrelevant or redundant features, which left us with 23 features from the 39 that we had initially. Before training the model, we decided to use *RandomUnderSampler* in the train data. As we know, using under sampling may increase the variance of the classifier and is very likely to discard useful samples. However, we still decided to work with this method because we had sufficient data and if the classes weren't balanced the model would just assign the predictions to the majority class and couldn't be considered a useful model, as we can see in the Figure 16. Keeping this in mind, our main goal was to have a good recall, precision and F1 score in the minority classes.

Again, we chose to evaluate each model by its F1 score, using the same reasoning. The models with the best F1 scores were *RandomForestClassifier*, *GradientBoostingClassifier*, *AdaBoostClassifier* and *HistGradientBoostingClassifier*. The best performing model ended up being the same one as in the binary classification, *GradientBoostingClassifier*, with a F1 Score of 0.61. Once again, the three final variables created in the feature engineering step were not used since they introduced data leakage.

5.2 Model assessment

To further optimize the chosen model's performance, a *GridSearch* was conducted to hyperparameter tune regarding the binary classification. Experimenting with the hyperparameters led the F1 score to reach a value of 0.62 on the training set, which shows that the model performs reasonably well. However, its performance on the test set gave a value of 0.35, which can be considered a drop in performance compared to the training set. Considering the given outcome, we reached the conclusion that hyperparameter tuning did not improve the performance of the model and, on top of that, we could suggest a potential issue of overfitting based on the high performance of the model on the training data. Thus, the decision was to use the default model of *GradientBoostingClassifier* without fine-tuning the hyperparameters.

Conclusion

In conclusion, this machine learning project has proved to be of great value to better understand the complexity of healthcare data. Despite the multiple challenges encountered, reflected on a modest F1 score of approximately 0.36 in the binary classification, and 0.61 in the multiclass classification, the models show potential in uncovering the factors that influence hospital readmissions for diabetic patients. However, contrary to the studies mentioned in the introduction, we did not find that MLP or Random Forests were the best predictors for our specific feature preprocessing and engineering, but Gradient Boosting Classifier.

When comparing the binary with the multiclass classification, it is evident that the latter revealed a better performance. Although we conducted some additional feature engineering for multiclass, there are other aspects that can have influenced these results. Usually, multiclass problems involve more complex relationships within the data, which can aid the model with decision boundaries and information gain. Moreover, feature importance might also have had an impact on this outcome since some variables can have been underutilized in the binary context. It's also important to recall that in the multiclass classification we prioritized having a practical model rather than an accurate model: while it is possible to get higher accuracy scores, this wouldn't necessarily be good from a business perspective. Regarding the binary classification, we believe that having more historical data and perhaps better variables could improve our predictions, therefore aiding in reducing healthcare costs.

On a personal level, we believe that this work was particularly meaningful to improve our understanding on data preprocessing, and how the steps taken before modeling deeply influence its performance. That said, it is important to keep in mind that a lot of optimizing steps, such as feature selection and hyperparameter fine-tuning, proved to be ineffective when it came to improving our predictions. All things considered, we can conclude that our work lays a solid foundation for a continuous effort to improve predictive algorithms in healthcare, serving as a starting point to enhance their practical implementation and significant impact.

References

- Goudjerkan, Ti'jay, and Manoj Jayabalan. "Predicting 30-Day Hospital Readmission for Diabetes Patients Using Multilayer Perceptron." International Journal of Advanced Computer Science and Applications, vol. 10, no. 2, 2019, https://doi.org/10.14569/ijacsa.2019.0100236. Accessed 24 Nov. 2019, https://researchonline.ljmu.ac.uk/id/eprint/11685/1/Paper_36-Predicting_30_Day_Hospital_Readmission_for_Diabetes_Patients.pdf

- "What Are Predictors of Medication Change and Hospital Readmission in Diabetic Patients? | UC Berkeley School of Information." Www.ischool.berkeley.edu, 2017, www.ischool.berkeley.edu/projects/2017/what-are-predictors-medication-change-and-hospital-readmission-diabetic-patients. Accessed 19 Dec. 2023,

  https://www.ischool.berkeley.edu/projects/2017/what-are-predictors-medication-change-and-hospital-readmission-diabetic-patients

- Hammoudeh, Ahmad, et al. "Predicting Hospital Readmission among Diabetics Using Deep Learning." Procedia Computer Science, vol. 141, 2018, pp. 484–489, https://doi.org/10.1016/j.procs.2018.10.138. Accessed 20 Apr. 2020,

  https://www.sciencedirect.com/science/article/pii/S1877050918317873?ref=pdf_download&fr=RR-2&rr=836a53a109db5bde

- "Winsorizing." The SAGE Encyclopedia of Educational Research, Measurement, and Evaluation, 2018, https://doi.org/10.4135/9781506326139.n747. Accessed 27 Mar. 2021 https://fisherpub.sjf.edu/statistics_facpub/7/

- "Box Cox Transformation: Definition, Examples." Statisticshowto.com, 20 Aug. 2021,

  www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/.

- Mazzanti, Samuele. "Boruta Explained the Way I Wish Someone Explained It to Me." Medium, 12 Feb. 2021,

  https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a

- Malato, Gianluca. "When and How to Use Power Transform in Machine Learning." Your Data Teacher, 21 Apr. 2021, www.yourdatateacher.com/2021/04/21/when-and-how-to-use-power-transform-in-machine-learning/

Appendix

Winsorizing

Outliers can disproportionately affect the mean and standard deviation of a dataset, leading to skewed distributions (as we had in our dataset). This technique limits the range of the data, thereby reducing the impact of outliers on these statistics and making the distribution more normally distributed.

Box-Cox

Having the distributions of numerical features as non-normal can have impacts on not being able to run a few numbers of tests, so applying box-cox is a correct approach to handle right skewness in a dataset as it makes it more symmetric, stabilizing the variance.

Boruta

This feature selection algorithm is thorough in identifying all relevant features because it's based on random forest, can capture non-linear relationships between features and it reduces the risk of overfitting.

PowerTransformer

A Power Transformer in Python applies monotonic transformations like Box-Cox to make data more Gaussian-like. This aids in addressing issues like non-constant variance or when normality is crucial for modeling. The transformer estimates optimal parameters for stability and skewness minimization via maximum likelihood. Box-Cox works with strictly positive data, while Yeo-Johnson handles both positive and negative values. The default behavior includes zero-mean, unit-variance normalization for standardized, Gaussian-shaped data.

Figures

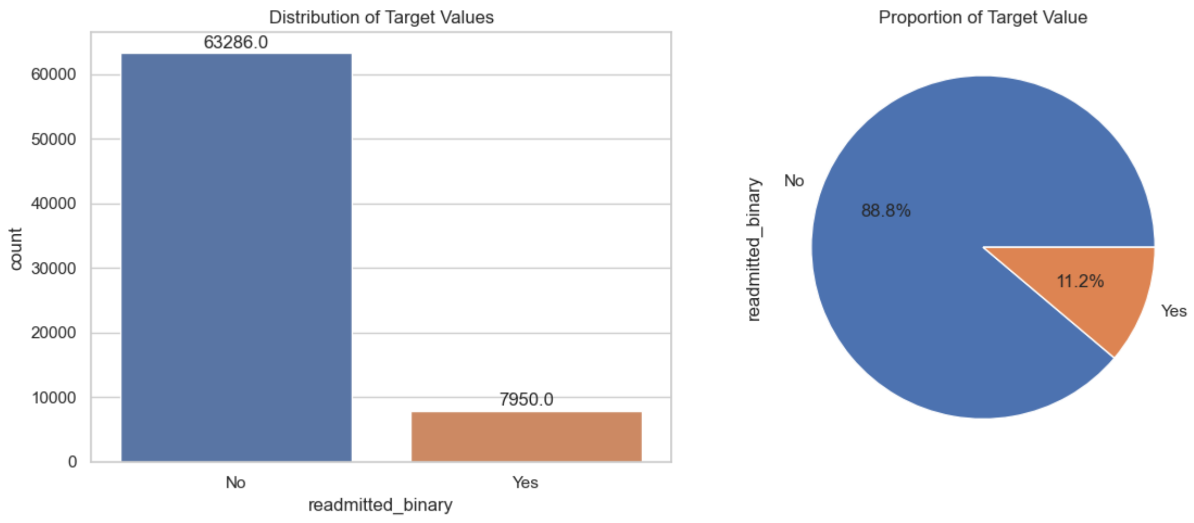
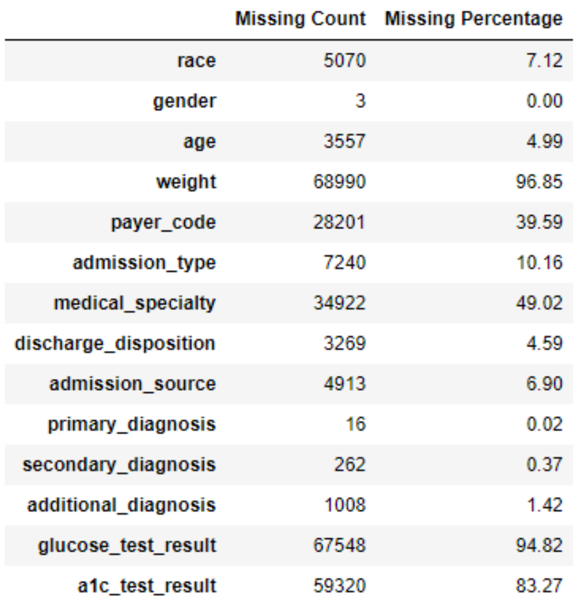


Figure 1 - Proportion of the target variable.



	Missing Count	Missing Percentage
race	5070	7.12
gender	3	0.00
age	3557	4.99
weight	68990	96.85
payer_code	28201	39.59
admission_type	7240	10.16
medical_specialty	34922	49.02
discharge_disposition	3269	4.59
admission_source	4913	6.90
primary_diagnosis	16	0.02
secondary_diagnosis	262	0.37
additional_diagnosis	1008	1.42
glucose_test_result	67548	94.82
a1c_test_result	59320	83.27

Figure 2 - Count and percentage of missing values.

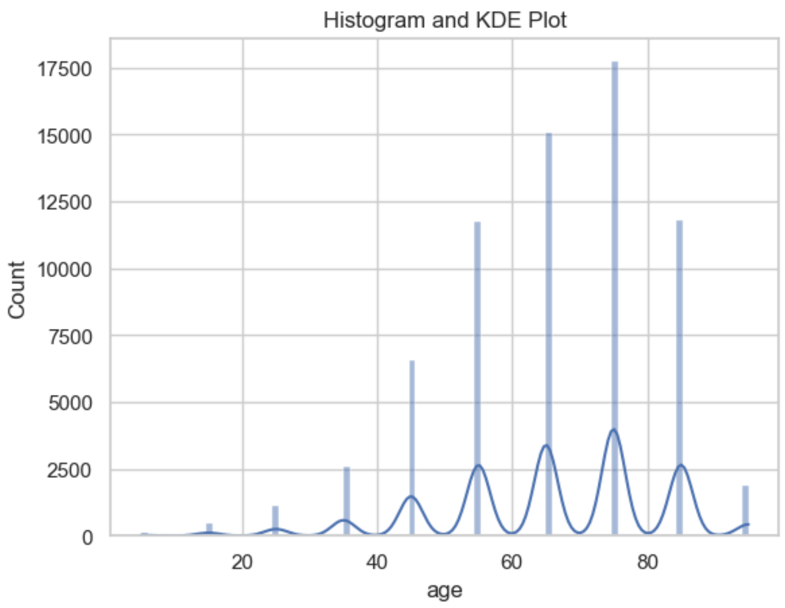


Figure 3 - Histogram and KDE plot of age.

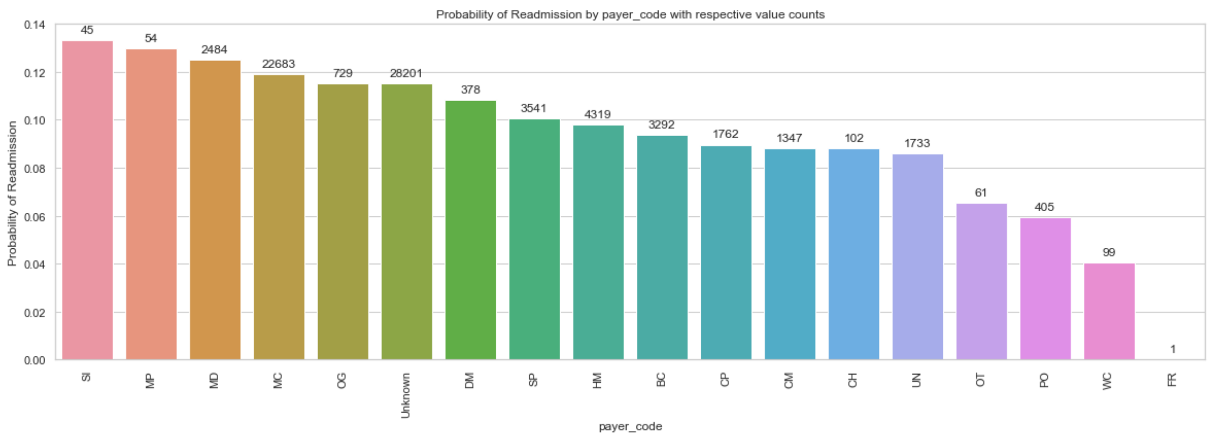


Figure 4 - Probability of readmission by payer_code with respective value counts.

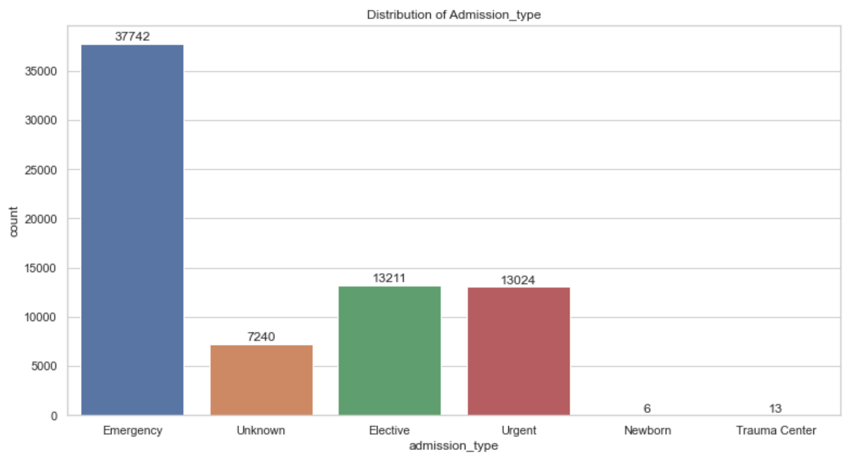
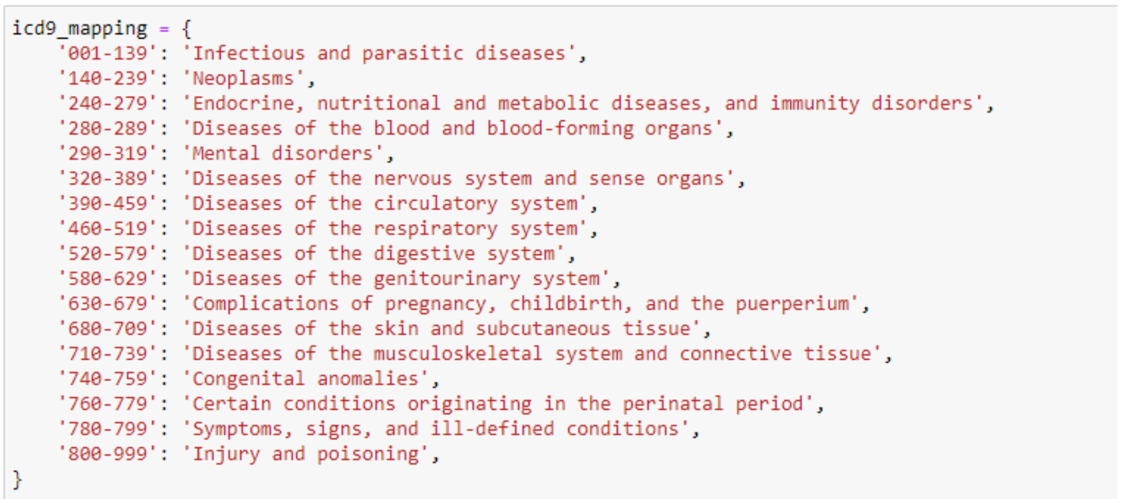


Figure 5 - Distribution of admission_type.



```
icd9_mapping = {
    '001-139': 'Infectious and parasitic diseases',
    '140-239': 'Neoplasms',
    '240-279': 'Endocrine, nutritional and metabolic diseases, and immunity disorders',
    '280-289': 'Diseases of the blood and blood-forming organs',
    '290-319': 'Mental disorders',
    '320-389': 'Diseases of the nervous system and sense organs',
    '390-459': 'Diseases of the circulatory system',
    '460-519': 'Diseases of the respiratory system',
    '520-579': 'Diseases of the digestive system',
    '580-629': 'Diseases of the genitourinary system',
    '630-679': 'Complications of pregnancy, childbirth, and the puerperium',
    '680-709': 'Diseases of the skin and subcutaneous tissue',
    '710-739': 'Diseases of the musculoskeletal system and connective tissue',
    '740-759': 'Congenital anomalies',
    '760-779': 'Certain conditions originating in the perinatal period',
    '780-799': 'Symptoms, signs, and ill-defined conditions',
    '800-999': 'Injury and poisoning',
}
```

Figure 6 - Mapping based on International Statistical Classification of Diseases and Related Health Problems.

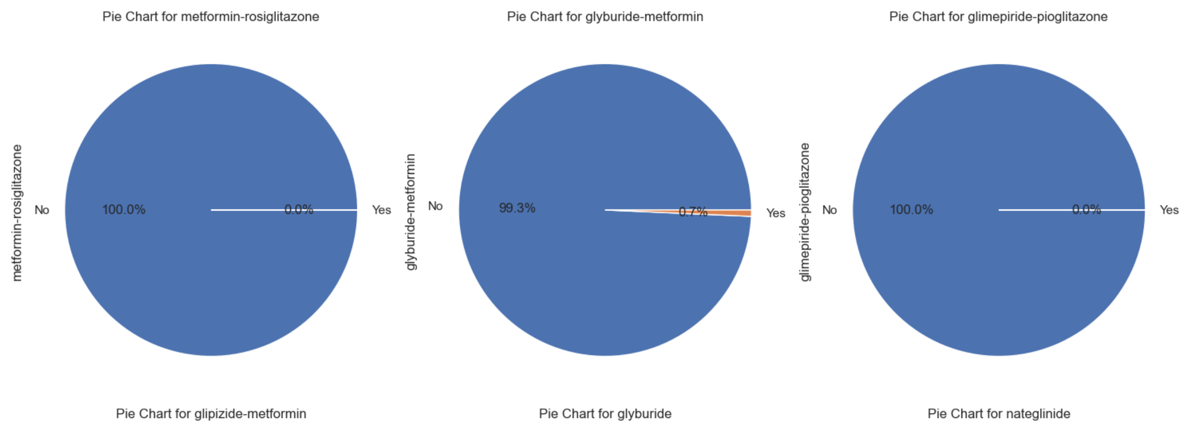


Figure 7 - Some examples of the plot that shows the prevalence of each group in medications.

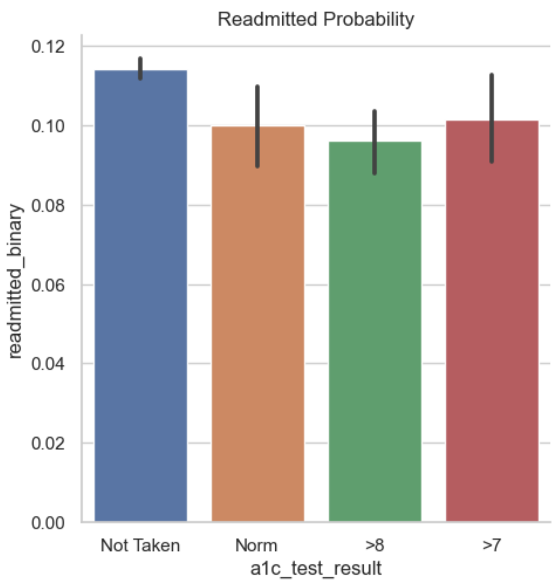
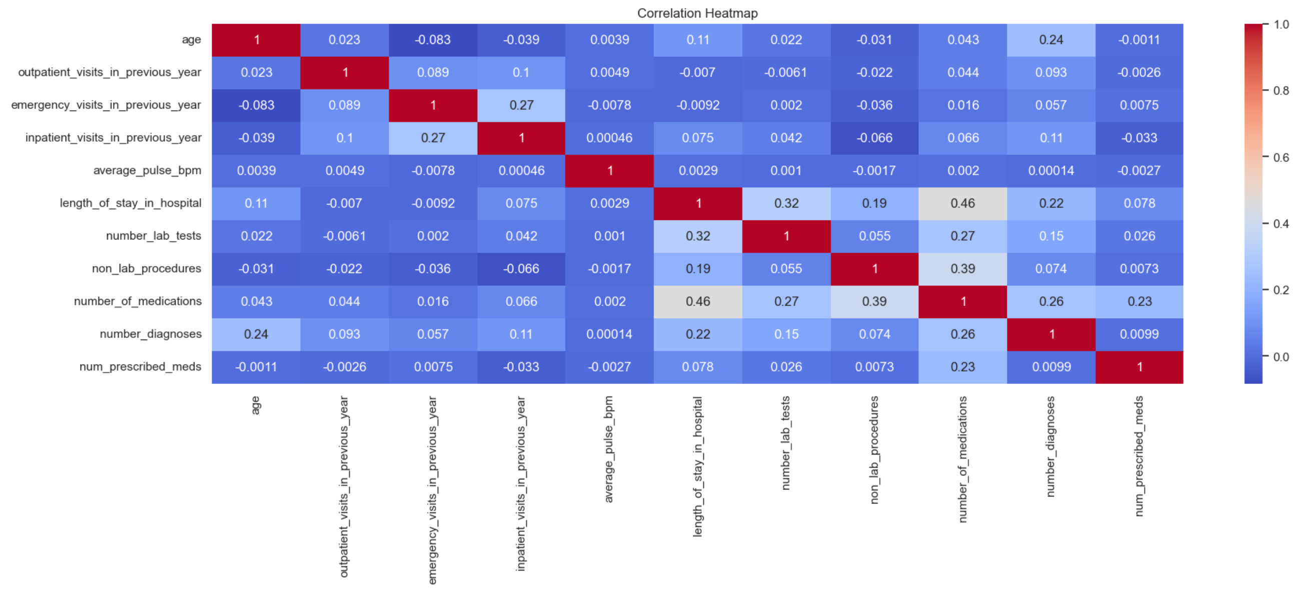


Figure 8 - Cat plot of a1c_test_results.



	age	outpatient_visits_in_previous_year	emergency_visits_in_previous_year	inpatient_visits_in_previous_year	average_pulse_bpm	length_of_stay_in_hospital	number_lab_tests	non_lab_procedures	number_of_medications	number_diagnoses	num_prescribed_meds
age	1	0.023	-0.083	-0.039	0.0039	0.11	0.022	-0.031	0.043	0.24	-0.0011
outpatient_visits_in_previous_year	0.023	1	0.089	0.1	0.0049	-0.007	-0.0061	-0.022	0.044	0.093	-0.0026
emergency_visits_in_previous_year	-0.083	0.089	1	0.27	-0.0078	-0.0092	0.002	-0.036	0.016	0.057	0.0075
inpatient_visits_in_previous_year	-0.039	0.1	0.27	1	0.00046	0.075	0.042	-0.066	0.066	0.11	-0.033
average_pulse_bpm	0.0039	0.0049	-0.0078	0.00046	1	0.0029	0.001	-0.0017	0.002	0.00014	-0.0027
length_of_stay_in_hospital	0.11	-0.007	-0.0092	0.075	0.0029	1	0.32	0.19	0.46	0.22	0.078
number_lab_tests	0.022	-0.0061	0.002	0.042	0.001	0.32	1	0.055	0.27	0.15	0.026
non_lab_procedures	-0.031	-0.022	-0.036	-0.066	-0.0017	0.19	0.055	1	0.39	0.074	0.0073
number_of_medications	0.043	0.044	0.016	0.066	0.002	0.46	0.27	0.39	1	0.26	0.23
number_diagnoses	0.24	0.093	0.057	0.11	0.00014	0.22	0.15	0.074	0.26	1	0.0099
num_prescribed_meds	-0.0011	-0.0026	0.0075	-0.033	-0.0027	0.078	0.026	0.0073	0.23	0.0099	1

Figure 9 - Correlation heatmap.

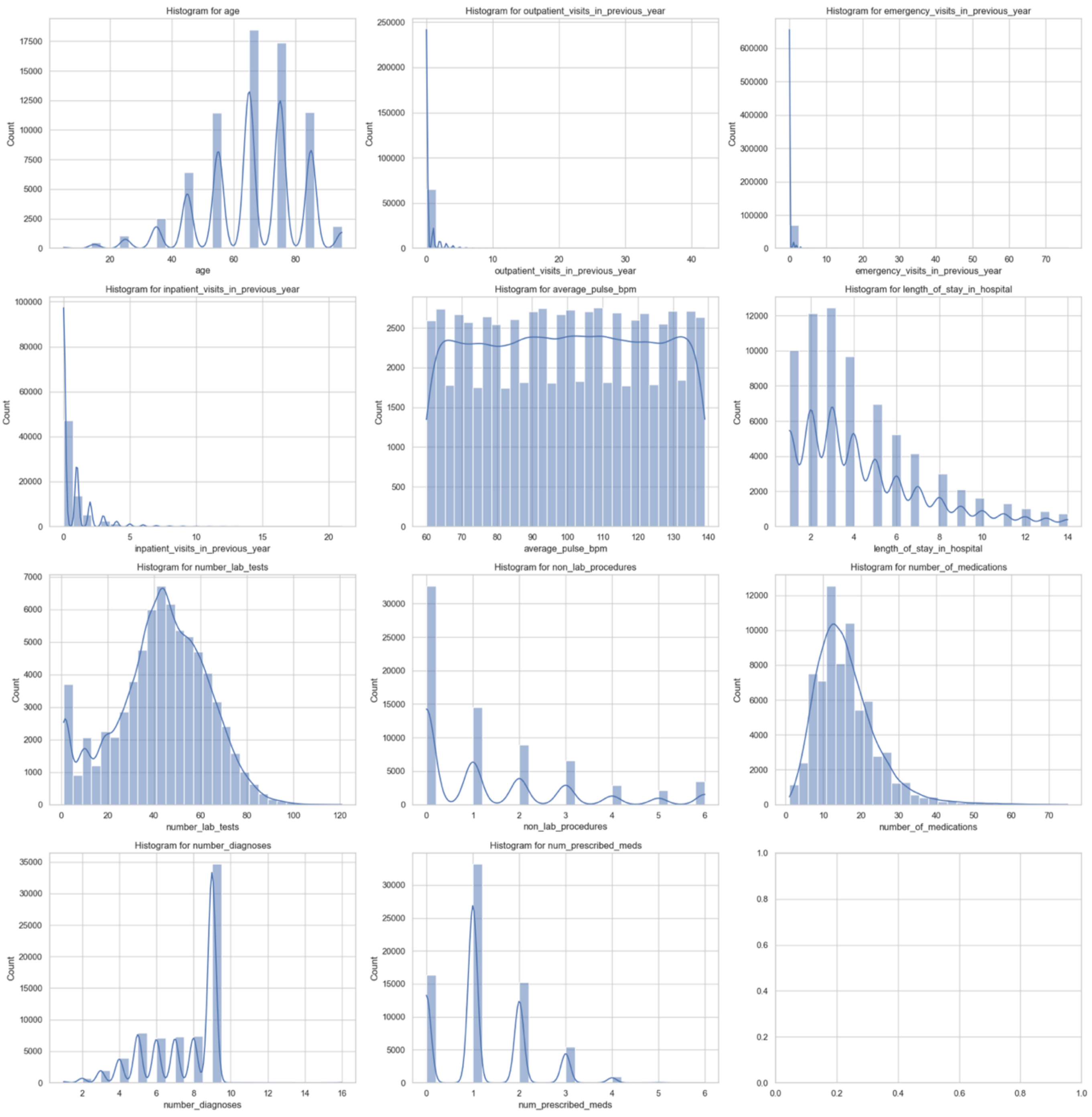


Figure 10 - Histograms of numeric features.

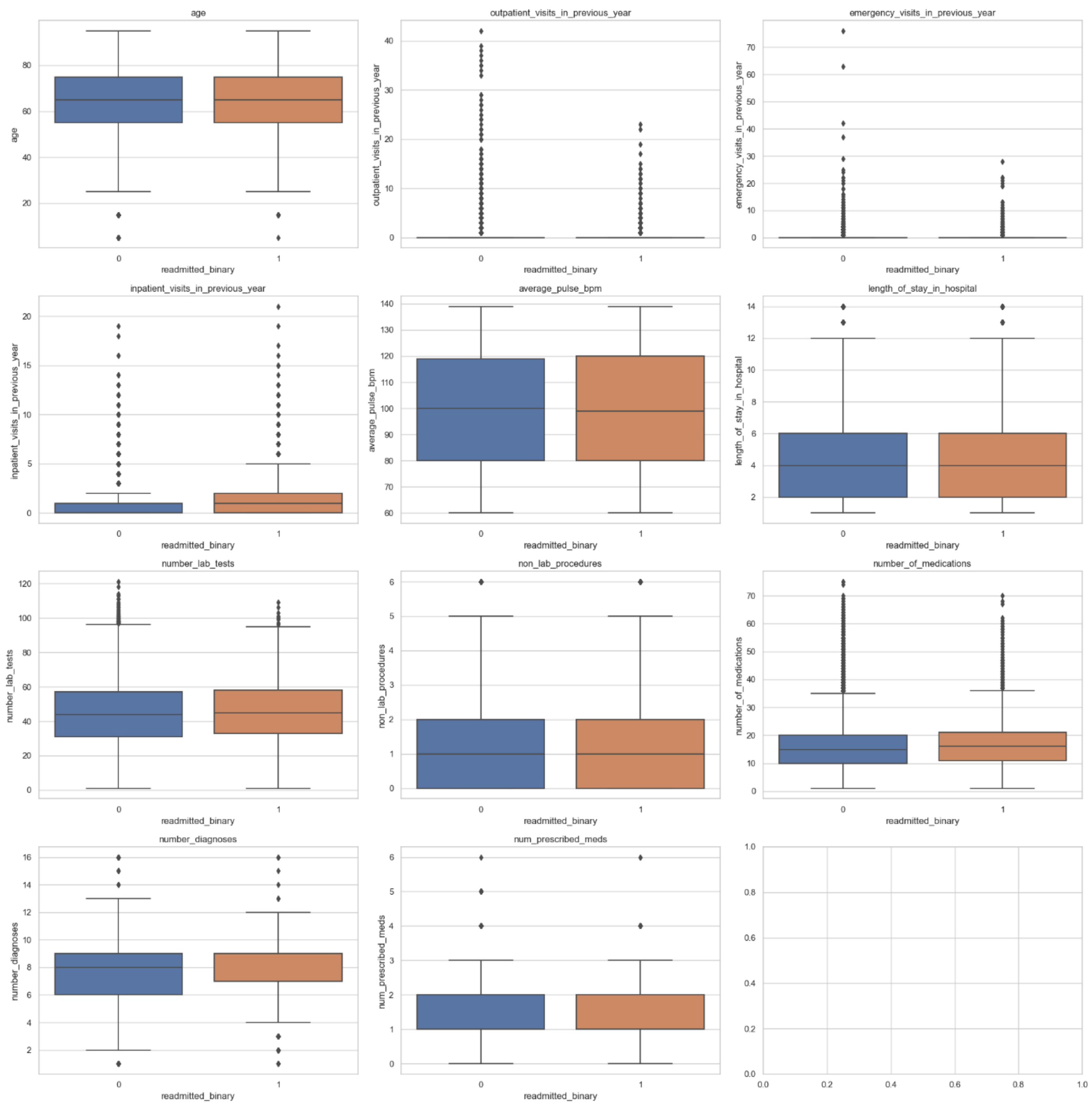


Figure 11 - Box plots of numerical features before applying winsorizing.

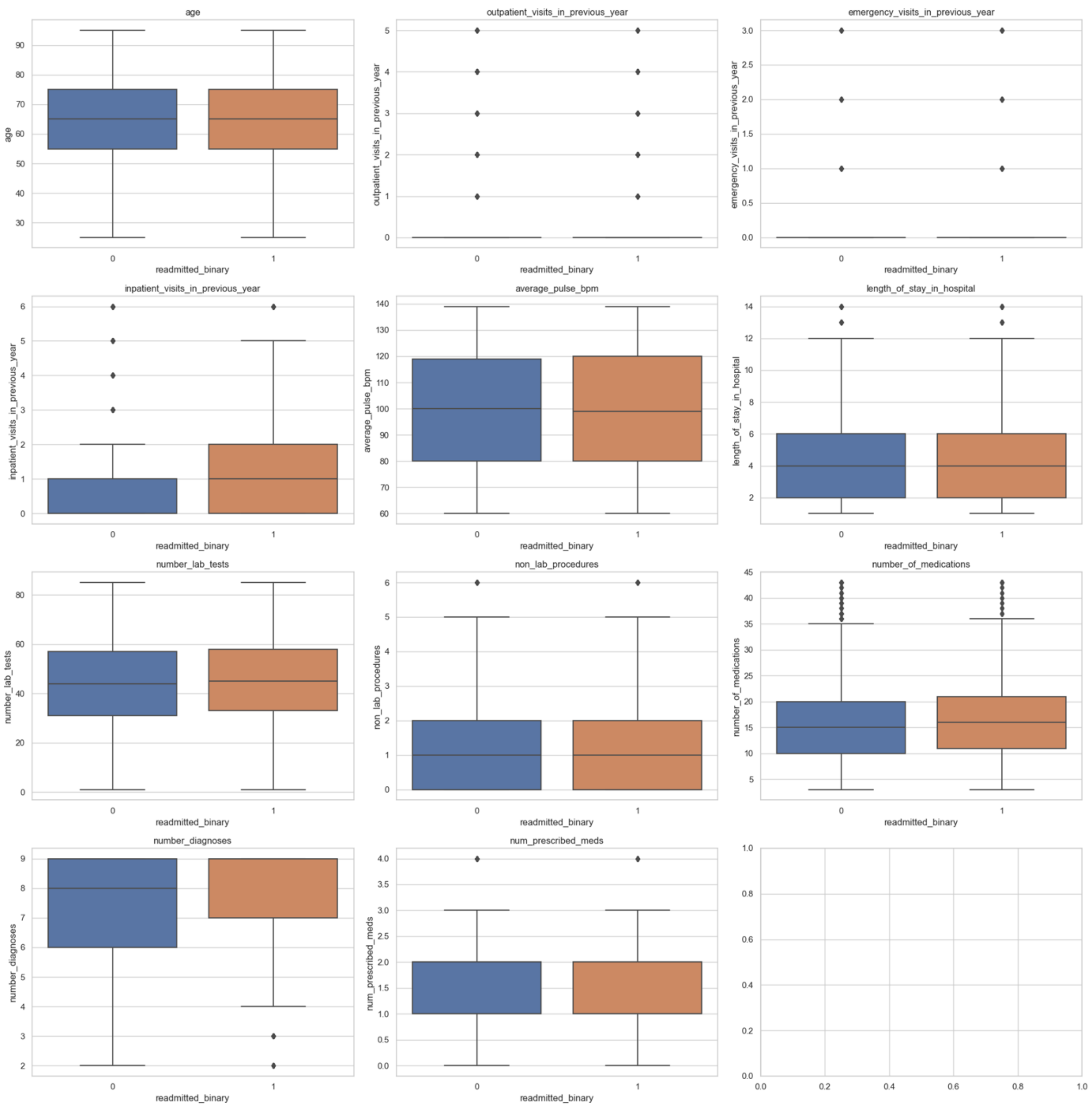


Figure 12 - Box plots of numerical features after applying winsorizing.

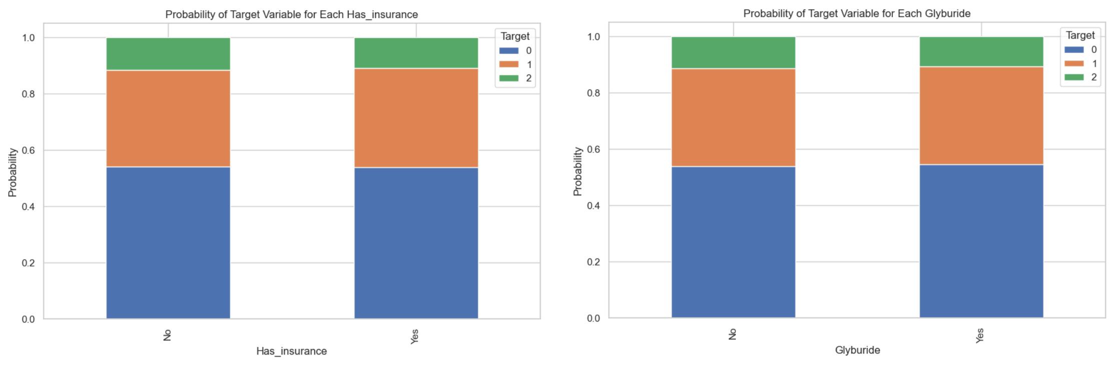


Figure 13 - Plot of the target variable distributions across groups in has_insurance and glyburide.

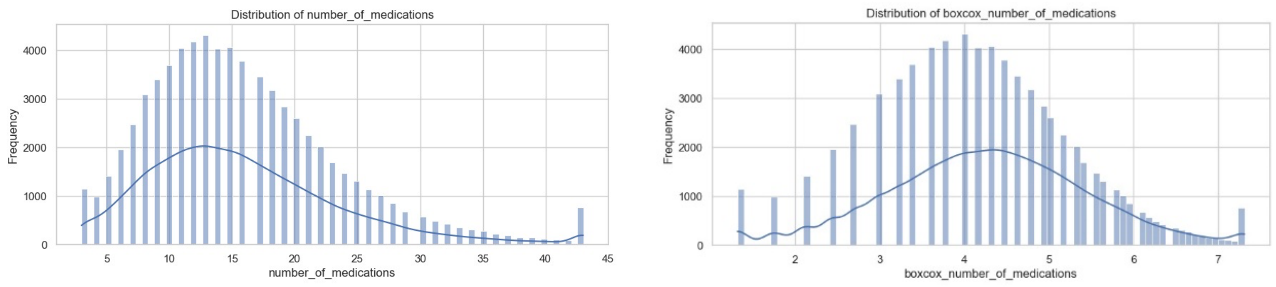


Figure 14 - Distribution of number_of_medications before and after box-cox transformation.

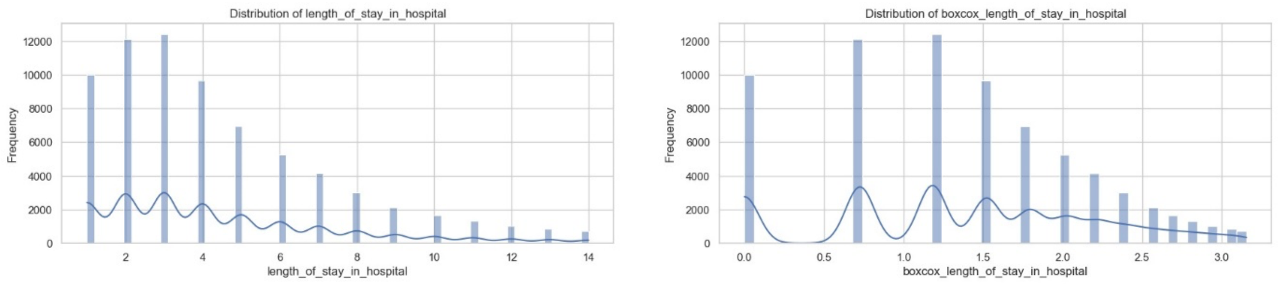
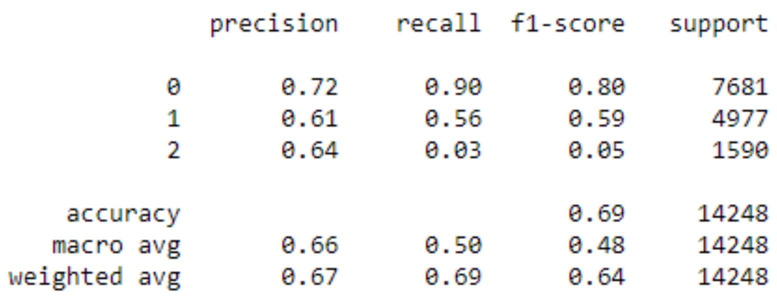


Figure 15 - Distribution of length_of_stay_in_hospital before and after box-cox transformation.



```
              precision    recall  f1-score   support

           0       0.72      0.90      0.80      7681
           1       0.61      0.56      0.59      4977
           2       0.64      0.03      0.05      1590

    accuracy                           0.69     14248
   macro avg       0.66      0.50      0.48     14248
weighted avg       0.67      0.69      0.64     14248
```

Figure 16 - Results without undersampling.