# NOVA IMS - Information Management School

Master's in data science and advanced Analytics, with specialization in Business Analytics

Machine Learning

1º Year // 1º Semester

**READY TO BE DISCHARGED:**
**EXAMINING HOSPITAL READMISSIONS**

Alexandre Spagnol

Ana Rita Silva

Guilherme Moreira

Jose Marçal

Ugochukwu Onyeri

Lisbon, 22 December of 2023

# NOVA IMS - Information Management School

Master's in data science and advanced Analytics, with specialization in Business Analytics

Machine Learning

1º Year // 1º Semester

**READY TO BE DISCHARGED:**
**EXAMINING HOSPITAL READMISSIONS**


**Students**: Alexandre Spagnol, nº 20230434
Ana Rita Silva, nº 20230444
Guilherme Moreira, nº 20230538
José Marçal, nº 20201581
Ugochukwu Onyeri, nº 20230817

**Professors**: Roberto Henriques
Ricardo Santos
Rafael Pereira


Lisbon, 22 December of 2023

# Index

## Table and figure index

## Abstract

Hospital readmissions within a short period after discharge are a significant issue in the healthcare sector, particularly for diabetic patients. These readmissions contribute substantially to healthcare costs, making it crucial to develop effective strategies to prevent them. This project aims to address this challenge by developing two machine-learning models for accurate hospital readmission prediction.

The two-pronged approach involves developing a binary classification model to predict whether a patient is likely to be readmitted within 30 days of discharge. This model provides a clear indication of readmission risk, allowing healthcare providers to take preventive measures and implement timely interventions. Additionally, a multiclass classification model is developed that classifies patients into three timeframes: no readmission, readmission within 30 days, and readmission beyond 30 days. This multi-class approach provides more nuanced insights into patient risk levels, allowing hospitals to tailor post-discharge care and follow-up procedures accordingly.

The data-driven approach utilizes two datasets: a training set and a testing set. The training set provides patient data encompassing demographic information, medical history, discharge diagnoses, and medication records, allowing machine learning algorithms to identify patterns and establish correlations between patient characteristics and readmission risks. The testing set independently evaluates the models' generalizability and robustness, assessing their ability to make accurate predictions in real-world healthcare applications.

Building upon proven techniques, the project utilizes machine learning algorithms such as random forests, decision trees, and neural networks. These algorithms have achieved promising results with decent rates for F1 Score (F-Measure) and Matthews Correlation Coefficient metric.

The project's main hypothesis is that the two machine learning models can accurately predict hospital readmissions, providing healthcare providers with valuable insights to improve patient care, reduce healthcare costs, and optimize resource allocation within the healthcare system.

The conclusions drawn from the results are that the two machine learning models can effectively predict hospital readmissions and provide valuable insights for healthcare providers to improve patient care, reduce healthcare costs, and optimize resource allocation within the healthcare system. The project's findings have the potential to impact healthcare practices and contribute to better patient outcomes significantly.

# Introduction

## Predicting hospital readmissions: A multiple-layer approach

Unplanned hospital readmissions within a short period after discharge pose significant challenges to healthcare systems, particularly for diabetic patients. These readmissions contribute substantially to healthcare expenditures, making it imperative to develop effective strategies to prevent them. This project addresses this challenge by employing machine learning to develop two classification models for accurate hospital readmission prediction (PSNet Patient Safety Network, 2019).

The primary objective of this project is to develop two machine-learning models that can effectively predict hospital readmissions. The first model will focus on identifying patients at high risk of readmission within 30 days, while the second model will classify patients into three timeframes: no readmission, readmission within 30 days, and readmission beyond 30 days (Casalini, et al., 2017).

## Prior research in hospital readmission prediction

The issue of hospital readmissions has been extensively investigated in medical literature. Previous studies have demonstrated the effectiveness of machine learning algorithms in predicting readmissions, particularly for specific patient populations such as those with diabetes.

Studies using machine learning algorithms like random forests, gradient boosting machines, and decision trees have achieved promising results in predicting 30-day readmissions among patients with various conditions, including heart failure and pneumonia. These studies have reported accuracy rates ranging from 65% to 75% and areas under the receiver operating curve (AUROC) measures of 0.65 to 0.78 (Huang, Talwar, Chatterjee, & Aparasu, 2021).

More recent research has explored the potential of machine learning to predict readmission timeframes beyond 30 days. Studies employing XGBoost algorithms have demonstrated the ability to accurately classify patients into multiple readmission timeframes, providing more granular insights into patient risk profile (Huang, Talwar, Lin, & Aparasu, 2022).

## Project expectations based on prior research

Building upon the findings of previous research, this project anticipates developing two machine-learning models with improved accuracy and generalizability compared to existing models. The binary classification model is expected to achieve an accuracy rate of at least 80%. The multiclass classification model is expected to achieve an accuracy rate of at least 75%. These models are expected to provide valuable insights to healthcare providers, enabling them to tailor post-discharge care and follow-up procedures for patients at varying risk of readmission. By effectively predicting readmissions and identifying high-risk patients early on, this project aims to contribute to improved patient outcomes, reduced healthcare costs, and optimized resource allocation within the healthcare system.

This project employs machine learning to develop two classification models for predicting hospital readmissions. The binary classification model focuses on identifying high-risk patients within 30 days, while the multi-class classification model provides detailed insights into readmission timeframes. These models have the potential to significantly improve patient care, reduce healthcare costs, and optimize resource allocation within the healthcare system.

## Data exploration and preprocessing

After importing the necessary libraries along with the training and test datasets, our first step was to perform some basic data exploration to check the statistics of each variable, like the mean and standard deviation for numerical variables, and the mode and unique values for categorical variables, as well as the correlation between each variable and the target variables. Afterwards, we checked the coherence of the dataset, by testing for the presence of duplicate entries (of which there were none) and missing values in each variable (present in the variables "race", "age", "admission_type", "discharge_disposition", "admission_source", "glucose_test_result" and "a1c_test_result", presented in ascending order for the number of missing values).

### Missing values

With the missing values identified, we first split the training dataset into a train and validation dataset before moving on to filling in the missing values according to our criteria. The lack of values in "glucose_test_result" and "a1c_test_result" indicates that these tests weren't needed, therefore we can assume that the patient's values were normal, and we substitute any missing value with "Norm". As for "race" and "age", given the small number of missing values, we decided to replace these with the mode for each given variable. Finally, "admission_type", "discharge_disposition" and "admission_source" already have a specific value named "Not Available", so any missing value was replaced with the former.

Our exploration of the data, however, also revealed that while some variables showed up as not having missing data, they were riddled with "?" values. These variables were "race", "payer_code", "medical_specialty" and "weight". For "race", the "?" values were taken care of the same way as the missing values were, by replacing them with the mode. As for "payer_code", we assumed that "?" represented the lack of insurance, so we substituted them for "No Insurance". Both "medical_specialty" and "weight" have a great amount of "?" and no clear way to replace them, so we decided to drop those columns.

### Outlier removal

Finally, before moving on to feature selection and engineering, we had to figure out the best way to remove outliers. To achieve this, we experimented with manual removal of the data, removing it according to the classic IQR definition of an outlier, removing it according to log transformations, and a method of combining both manual and IQR removal. In the end, we decided the manual method was better, as it kept the most amount of data to be used.

### Feature selection and engineering

We first performed some data optimization by encoding the binary variables "change_in_meds_during_hospitalization" and "prescribed_diabetes_meds" and by converting all variables of data type int64 to data type int16 to save space and memory. With that step done, our first mission was to change the "age" value bins, by transforming all ages in the [0-10) bracket to "child", all ages in the [10-40) bracket to "young", all ages in the [40-70) bracket to "middle", and any age above 70 to "old".

Our next mission was to create new features to add to our dataset, where we created the
"total_visits_in_previous_year" (the sum of inpatient, outpatient and emergency visits),
"total_procedures_during_stay" (the sum of lab tests and procedures),
"medications_per_day_of_stay" (ratio of number of medication and length of stay in hospital),
"procedures_per_day_of_stay" (ratio of number of total procedures and length of stay in hospital),
"procedures_diagnoses_ratio" (ratio of total procedures and diagnoses in hospital) and

"number_of_visits" (sum of how many times a patient_id comes up, counting the visits that patient made in total).

Next up, we decided to encode the categorical variables, given some models require only numerical inputs to work. After testing with both label and ordinal encoders, we decided that the label encoder would be the most appropriate. To end the feature engineering process, we also scaled our numerical variables, since some models provide inaccurate results with unscaled data. For this process, we tested a MinMax scaler, a MinMax (-1, 1) scaler, a Standard scaler, and a Robust scaler, before concluding that the Robust scaler was the best solution (Figure 6, Figure 5).

To finish up the preprocessing part, it was time to do the feature selection. Given that "country" only has a single value, we decided to drop that column without any further analysis. Regarding the categorical variables, we performed a Chi-Square test with 95% significance, where we concluded we should also drop "race", "gender" and "admission_type". As for the numerical variables, we tested their Pearson correlation and performed RFE with 3 separate evaluation models (Linear Regression, Decision Tree, and Gradient Boost Classifier) and a Lasso Regression. Taking the results of each model into account, we decided whether we should keep each variable (Table 1).

Every model or metric used during this stage will have the documentation linked at the end.

| Predictor | Pearson | RFE LR | RFE GBC | RFE DT | LassoCV | Chi-Square | Decisons |
|---|---|---|---|---|---|---|---|
| outpatient_visits_in_previous_year | Drop | Keep | Drop | Drop | Keep | - | Drop |
| emergency_visits_in_previous_year | Drop | Keep | Drop | Drop | Keep | - | Drop |
| inpatient_visits_in_previous_year | Drop | Keep | Keep | Drop | Keep | - | Keep |
| average_pulse_bpm | Keep | Drop | Drop | Keep | Drop | - | Keep |
| length_of_stay_in_hospital | Keep | Drop | Drop | Drop | Drop | - | Drop |
| number_lab_tests | Drop | Drop | Drop | Keep | Drop | - | Drop |
| non_lab_procedures | Keep | Drop | Drop | Drop | Drop | - | Drop |
| number_of_medications | Keep | Drop | Keep | Keep | Keep | - | Keep |
| number_diagnoses | Keep | Keep | Keep | Drop | Keep | - | Keep |
| change_in_meds_during_hospitalization | Keep | Drop | Drop | Drop | Drop | - | Drop |
| prescribed_diabetes_meds | Keep | Drop | Drop | Drop | Drop | - | Drop |
| total_visits_in_previous_year | Keep | Keep | Keep | Drop | Drop | - | Keep |
| total_procedures_during_stay | Keep | Drop | Keep | Drop | Drop | - | Drop |
| medication_per_day_of_stay | Keep | Keep | Drop | Keep | Keep | - | Keep |
| procedures_per_day_of_stay | Keep | Drop | Keep | Keep | Keep | - | Keep |
| procedure_diagnosis_ratio | Keep | Drop | Drop | Keep | Drop | - | Keep |
| number_of_vistis | Keep | Keep | Keep | Keep | Keep | - | Keep |
| race | - | - | - | - | - | Drop | Drop |
| gender | - | - | - | - | - | Drop | Drop |
| age | - | - | - | - | - | Keep | Keep |
| payer_code | - | - | - | - | - | Keep | Keep |
| admission_type | - | - | - | - | - | Drop | Drop |
| discharge_disposition | - | - | - | - | - | Keep | Keep |
| admission_source | - | - | - | - | - | Keep | Keep |
| primary_diagnosis | - | - | - | - | - | Keep | Keep |
| secondary_diagnosis | - | - | - | - | - | Keep | Keep |
| additional_diagnosis | - | - | - | - | - | Keep | Keep |
| glucose_test_result | - | - | - | - | - | Keep | Keep |
| a1c_test_result | - | - | - | - | - | Keep | Keep |
| medication | - | - | - | - | - | Keep | Keep |

*Table 1. Feature Selection Decision*

## Binary classification

### Additional preprocessing

Given the big imbalance of the target variable (readmitted_binary) where 89% (63286 rows) of the data is 'No' and only 11% (7950 rows) of the data is 'Yes', we needed to come up with a solution to combat this imbalance, otherwise this could lead to misleadingly high accuracies or F1 scores.

Therefore, to resist this highly imbalanced target variable we implemented resampling techniques. For this we had two options, undersampling or oversampling. Both have their pros and cons, but ultimately, we chose oversampling, more specifically SMOTE, for three main reasons. Firstly, if we decided to use undersampling, we would be removing observations from our dataset and consequently losing precious information, which would reduce the amount of data our models had to train therefore possibly leading to underfitting. Secondly, we believed that the initially provided 7950 entries would not be enough to properly train and accurately predict results for the 'Yes' target variable, which meant we had to artificially insert data using oversampling. Finally, there is a study reinforcing all the points given previously, by stating *"Overall, the hybrid resampling approach (SMOTE) seems to work well in an extremely imbalanced condition, whereas RandomOverSampling is not recommended due to its vulnerability to overfitting. Across both moderately and extremely imbalanced cases, RandomUnderSampling is not recommended as the first choice as it leads to the loss of potentially useful data for the classifier"* (Wongvorachan et al. #). We were aware that oversampling often increases the chance of overfitting, which we were affected by, but it improved our results.

### Modelling approach

Regarding model assessment, we used the StratifiedKFold function with 10 splits to evaluate our models. This model follows the cross-validation technique by randomly shuffling the dataset and then splitting it into K (10) groups/folds. The 'Stratified' part is regarding the target variable, where the folds are made by preserving the percentage of target variable samples. We also divided our dataset into two partitions, them being the training (80%) and the validation/holdout (20%) partitions. This way, we can successfully train our models with enough data and still be able to test our predictions with unused rows.

To fully understand the weight hyperparameters had in our predictions, we initially called the models with their default values (Table 2). Later, we started to fine-tune our models by modifying their hyperparameters. To do this, we adopted a combination of the GridSearchCV algorithm paired with some manual adjustments where we deemed necessary, often paired with visual representations (using pyplot) to better understand the results.

After some testing, we came up with the models that provided the best results, distributed across different categories (Linear, Tree-Based, Ensemble, etc.). These models were LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, KNNClassifier, GaussianNaiveBayes, Multi-Layer Perceptrons and Voting/Majority Rule.

## Performance assessment

To maintain consistency and reproducibility across the different models, we used the *random_state* parameter. Reproducibility is crucial as it allows us to recreate the same random conditions for each experiment, helping us create a level playing field for comparison. This consistency ensures that any differences in performance are more likely attributed to the model's characteristics rather than randomness we cannot control.

Concerning performance assessment, we resorted to various metrics such as accuracy, precision, recall, ROC AUC, and F1 Score. Due to the nature of the dataset being imbalanced and various papers defending the F1 Score (F-Measure) as a highly popular metric for this kind of data (Brownlee) (Brownlee) (Bajaj) and also the Matthews Correlation Coefficient as "*the best choice"* for this kind of data (Heras, Martín, Carrasco, & Luque, 2019), we decided to use them as our main metrics. We used two branches of the F1 Score, the balanced and the weighted. The balanced F1 Score was used to prevent the model from being biased toward the majority class, providing a more comprehensive measure of overall model performance. On the other hand, we used the weighted F1 Score to consider the imbalance and assign a higher weight to the class with more instances, as we believed it was more important to correctly predict people who wouldn't be readmitted to the hospital. This ensures the model's performance on the larger class has a more significant impact on the overall score.

## Final thoughts

After training all the models with the oversampled dataset and the best ideal hyperparameters, the model that performed the best (ranked by the Matthews Correlation Coefficient and balanced F1 Score) was RandomForestClassifier with a Matthews Correlation Coefficient of 0.168 and F1 Score of 0.280, closely followed by DecisionTreeClassifier with a MCC of 0.161 and F1 Score of 0.275. Ranked third place was LogisticRegression with an MCC of 0.137 and an F1 Score of 0.248 (Table 3). We believe random forests ranked highly due to their robustness against overfitting (which we suffer a lot from) and the complexity of the dataset (not linear). It's also important to consider the other metrics. Every model or metric used during this stage will have the documentation linked at the end.

| Default Models | F1-Score (Weighted) | Accuracy | Precision | Recall | AUC | Matthews | F1-Score (Default) |
|---|---|---|---|---|---|---|---|
| Logistic Regression | 0,811 | 0,5795 | 0,8294 | 0,796 | 0,5795 | 0,1217 | 0,2475 |
| MLPClassifier | 0,7675 | 0,577 | 0,8266 | 0,7273 | 0,577 | 0,0935 | 0,2389 |
| KNN Classifier | 0,7725 | 0,5433 | 0,8159 | 0,7396 | 0,5433 | 0,072 | 0,1993 |
| Voting Classifier | 0,8406 | 0,533 | 0,8241 | 0,8666 | 0,5331 | 0,028 | 0,1479 |
| DecisionTreeClassifier | 0,8063 | 0,509 | 0,8053 | 0,8072 | 0,509 | 0,0881 | 0,1266 |
| RandomForestClassifier | 0,8363 | 0,5013 | 0,8164 | 0,8874 | 0,5013 | 0,0973 | 0,0086 |
| GaussianNB | 0,8359 | 0,5 | 0,7893 | 0,8884 | 0,5 | 0 | 0 |

*Table 2. Default Benchmark for Binary Classification*

| Tuned Models | F1-Score (Weighted) | Accuracy | Precision | Recall | AUC | Matthews | F1-Score (Default) |
|---|---|---|---|---|---|---|---|
| RandomForestClassifier | 0,7933 | 0,6105 | 0,8378 | 0,7624 | 0,6105 | 0,1684 | 0,2805 |
| DecisionTreeClassifier | 0,7901 | 0,6067 | 0,8365 | 0,758 | 0,6067 | 0,1616 | 0,2753 |
| MLPClassifier | 0,7805 | 0,6081 | 0,8367 | 0,7434 | 0,6081 | 0,1326 | 0,274 |
| Logistic Regression | 0,8113 | 0,5797 | 0,8295 | 0,7961 | 0,5798 | 0,1372 | 0,248 |
| KNN Classifier | 0,8006 | 0,5386 | 0,8154 | 0,7874 | 0,5386 | 0,0681 | 0,1864 |
| Voting Classifier | 0,8402 | 0,5336 | 0,8238 | 0,8655 | 0,5336 | 0,1019 | 0,1499 |
| GaussianNB | 0,8359 | 0,5 | 0,7893 | 0,8884 | 0,5 | 0 | 0 |

*Table 3. Optimized Models for Binary Classification*

## Multiclass classification

### Additional preprocessing

The preprocessing made for multiclass classification, specifically was not very different from the one made for binary classification. The data was imbalanced, so balancing was needed, and we came to the same conclusion as in binary, SMOTE was the superior technique. We tried removing some features that the model RandomForestClassifier was not giving much importance to, but since it resulted in a lower score, we decided not to do it.

### Modelling approach

To perform multiclass classification, we used as our models Logistic Regression, RandomForestClassifier, SupportVectorsClassifier's, GaussianNaiveBayes, KNN Classifier and DecisionTreeClassifier.

We initially called our models with their default values to test their hyperparameters later (Table 4).

We applied hyperparameter tuning in almost every model, except for neural networks, where we used the same model as the one for binary. To find the best parameters, we also did a mix of different GridSearchCV and manual search.

To define our optimal model, we also established thresholds of 0.40 and 0.55 for the F1 training score and validation score, respectively. These criteria led us to select the RandomForestClassifier and SVC as our best models for multiclass classification. However, even though our overall F1 scores were very similar between the two models, both achieving a validation score of 0.55 while doing a more detailed comparison, the SVC outperformed the RandomForestClassifier by 0.003 in the validation score.

While analyzing the detailed metrics, we observed that the RandomForestClassifier exhibited slightly better validation scores in terms of class distribution and the remaining metrics (Table 5).

So, based on these considerations, we chose the RandomForestClassifier as our optimal model for Multiclass Classification.

### Performance assessment

To maintain consistency and reproducibility across the different models, we used the *random_state* parameter. Reproducibility is crucial as it allows us to recreate the same random conditions for each experiment, helping us create a level playing field for comparison. This consistency ensures that any differences in performance are more likely attributed to the model's characteristics rather than randomness we cannot control, just like we did in binary classification.

In a nutshell, the performance assessment is like the one in binary classification, except for ROC AUC, because unfortunately we were not able the find a way to make it work. We still managed to evaluate the model's score using the rest of the metrics. The regular f1 score is not valid for multiclass classification, so we just used the weighted version of it, which will consider the imbalanced weights on the validation dataset, as we believed it was more important to correctly predict people who wouldn't be readmitted to the hospital. This ensures the model's performance on the larger class has a more significant impact on the overall score.

## Final thoughts

After training all the models with the oversampled dataset and the best ideal hyperparameters, the model that performed the best (ranked by the Matthews Correlation Coefficient and weighted F1 Score) was RandomForestClassifier with a Matthews Correlation Coefficient of 0.243 and F1 Score of 0,55, closely followed by SVC with a MCC of 0.225 and F1 Score of 0.553. Ranked third place was NuSVC with an MCC of 0.216 and an F1 Score of 0.541. We believe random forests ranked highly due to their robustness against overfitting (which we suffer a lot from) and the complexity of the dataset (not linear). It's also important to consider the other metrics. We can safely say that there is no clear winner, just slightly better ones. Also, if time was not so short, we would invest more time in SVM classifiers because they seemed to perform well with an almost default setting. Every model or metric used during this stage will have the documentation linked at the end.

| Default Models | F1-Score (Weighted) | Accuracy | Precision | Recall | Matthews |
|---|---|---|---|---|---|
| RandomForestClassifier | 0,5521 | 0,4361 | 0,5511 | 0,5846 | 0,2281 |
| MLPClassifier | 0,5312 | 0,4381 | 0,5517 | 0,5571 | 0,2108 |
| Logistic Regression | 0,489 | 0,4201 | 0,5668 | 0,5669 | 0,2046 |
| KNN Classifier | 0,4769 | 0,412 | 0,5107 | 0,4565 | 0,1294 |
| DecisionTreeClassifier | 0,4599 | 0,3761 | 0,4738 | 0,4493 | 0,079 |
| Voting Classifier | 0,455 | 0,4343 | 0,5573 | 0,4736 | 0,166 |
| GaussianNB | 0,0224 | 0,3333 | 0,0125 | 0,1116 | 0 |
| SVC | 0,5361 | 0,4414 | 0,5671 | 0,5797 | 0,2356 |
| NuSVC | 0,5413 | 0,4402 | 0,5478 | 0,5621 | 0,2159 |

*Table 4. Default Benchmark for Multiclass Classification*

| Tuned Models | F1-Score (Weighted) | Accuracy | Precision | Recall | Matthews |
|---|---|---|---|---|---|
| RandomForestClassifier | 0,5503 | 0,4534 | 0,5649 | 0,5839 | 0,2433 |
| MLPClassifier | 0,4102 | 0,4305 | 0,3548 | 0,5089 | 0,1666 |
| Logistic Regression | 0,4888 | 0,4199 | 0,5665 | 0,5667 | 0,2044 |
| KNN Classifier | 0,4967 | 0,4355 | 0,5359 | 0,4808 | 0,1681 |
| DecisionTreeClassifier | 0,527 | 0,4255 | 0,5637 | 0,5761 | 0,2221 |
| Voting Classifier | 0,4102 | 0,4387 | 0,5677 | 0,466 | 0,1587 |
| GaussianNB | 0,0224 | 0,3333 | 0,0125 | 0,1116 | 0 |
| SVC | 0,5537 | 0,4364 | 0,5471 | 0,5766 | 0,2248 |
| NuSVC | 0,5413 | 0,4402 | 0,5478 | 0,5621 | 0,2159 |

*Table 5. Optimized Models for Multiclass Classification*

## Conclusion

In conclusion, the project provided valuable insights into predicting hospital readmissions with a decent level of certainty. These types of projects result in advantages for hospitals that perform them. Advantages start from social to economic ones.

Hospitals can understand and flag possible returning patients and act proactively, asking for example certain types of patients to return before those 30 days, or even simple things like checkup calls.

We expected the accuracy level to be higher, but based on the type and the quality of the data we were given, the results were very good. A few things we could have done better start with preprocessing and finish with a broader portfolio of models.

Regarding preprocessing, we could have invested more in feature engineering and the development of new features. On the model side, maybe more fine-tuning could have resulted in better results, but time was precious and hyper tuning is very time-consuming. We can say we are happy with our work.

Some limitations we found were around the data, the imbalance dataset, and mainly some features that could have been useful were dropped due to lack of values.

Regarding follow-up work, we could get more specific values for some features like weight, and age. Also, new features related to basic medical information like height could have been valuable. Finally, try and get more data for the minority classes, to fix the imbalance, more research needs to be carried out on this topic to gain more insight.

Triple bam!

## Bibliography


Bajaj, A. (2023, 12 8). *Performance Metrics in Machine Learning [Complete Guide] - neptune.ai.* Retrieved 12 2023, from Neptune.ai: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide#h-f1-score

Brownlee, J. (2020, August 2). *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification - MachineLearningMastery.com.* Retrieved 2023, from Machine Learning Mastery: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/#AdThrive_Content_10_desktop

Brownlee, J. (2021). *Tour of Evaluation Metrics for Imbalanced Classification.* Retrieved from MachineLearningMastery: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/#AdThrive_Content_5_desktop

Casalini, F., Salvetti, S., Memmini, S., Memmini, S., Lucaccini, E., Massimetti, G., . . . Privitera, G. P. (2017). International Journal for Quality in Health Care. *Unplanned readmissions within 30 days after discharge: improving quality through easy prediction*, 256–261.

Heras, A. d., Martín, A., Carrasco, A., & Luque, A. (2019, February 22). *The impact of class imbalance in classification performance metrics based on the binary confusion matrix*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S0031320319300950#abs0001

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 Med Res Methodol* .

Huang, Y., Talwar, A., Lin, Y., & Aparasu, R. R. (2022). Machine learning methods to predict 30-day hospital readmission outcome among US adults with pneumonia: analysis of the national readmission database. *BMC Medical Informatics and Decision Making*.

PSNet Patient Safety Network. (2019, September 7). *Readmissions and Adverse Events After Discharge*. Retrieved from PSNet Patient Safety Network: https://psnet.ahrq.gov/primer/readmissions-and-adverse-events-after-discharge

Scikit-learn developers. (n.d.). *1.12. Multiclass and multioutput algorithms*. Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/multiclass.html#multiclass-classification

Scikit-learn developers. (n.d.). *3.3. Metrics and scoring: quantifying the quality of predictions*. Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/model_evaluation.html#binary-classification

Scikit-learn developers. (n.d.). *User Guide*. Retrieved from Scikit-learn: https://scikit-learn.org/stable/user_guide.html

Wongvorachan, T., He, S., & Bulut, O. (2023). A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining. Retrieved from Information: https://www.mdpi.com/2078-2489/14/1/54#sec6-information-14-00054

## Annexes

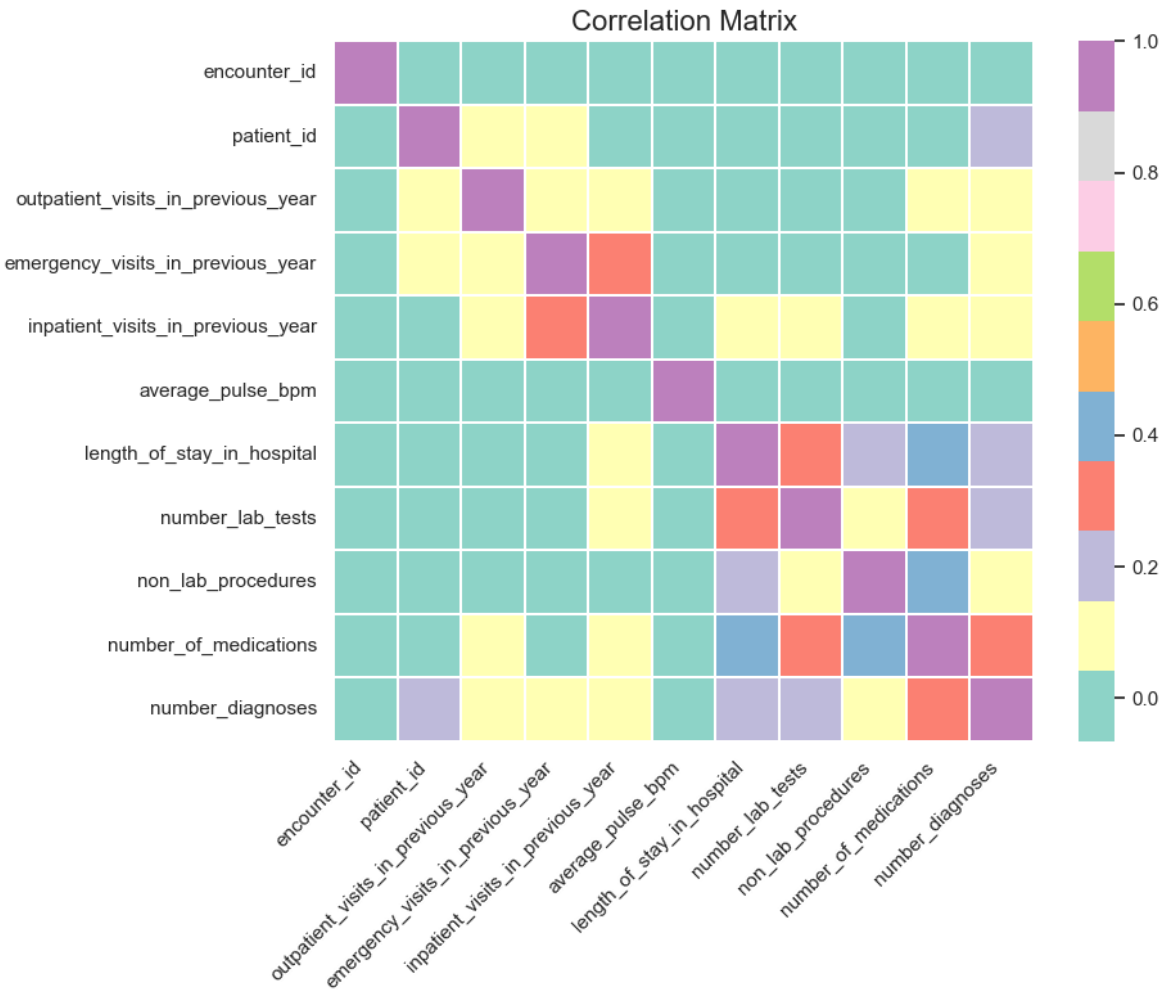


*Figure 1. Correlation Matrix*

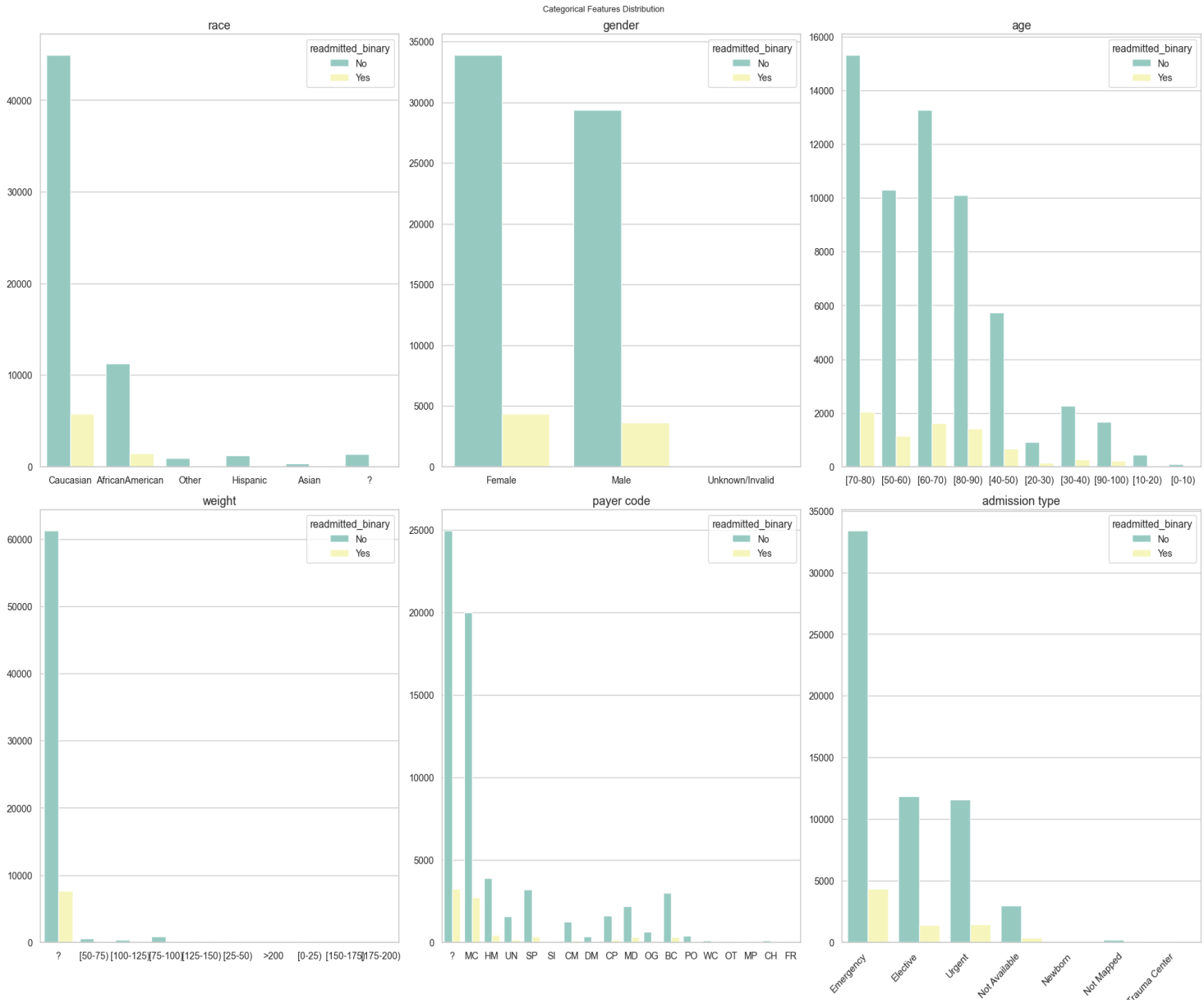


*Figure 2. Categorical Feature Relation to Imbalanced Target*

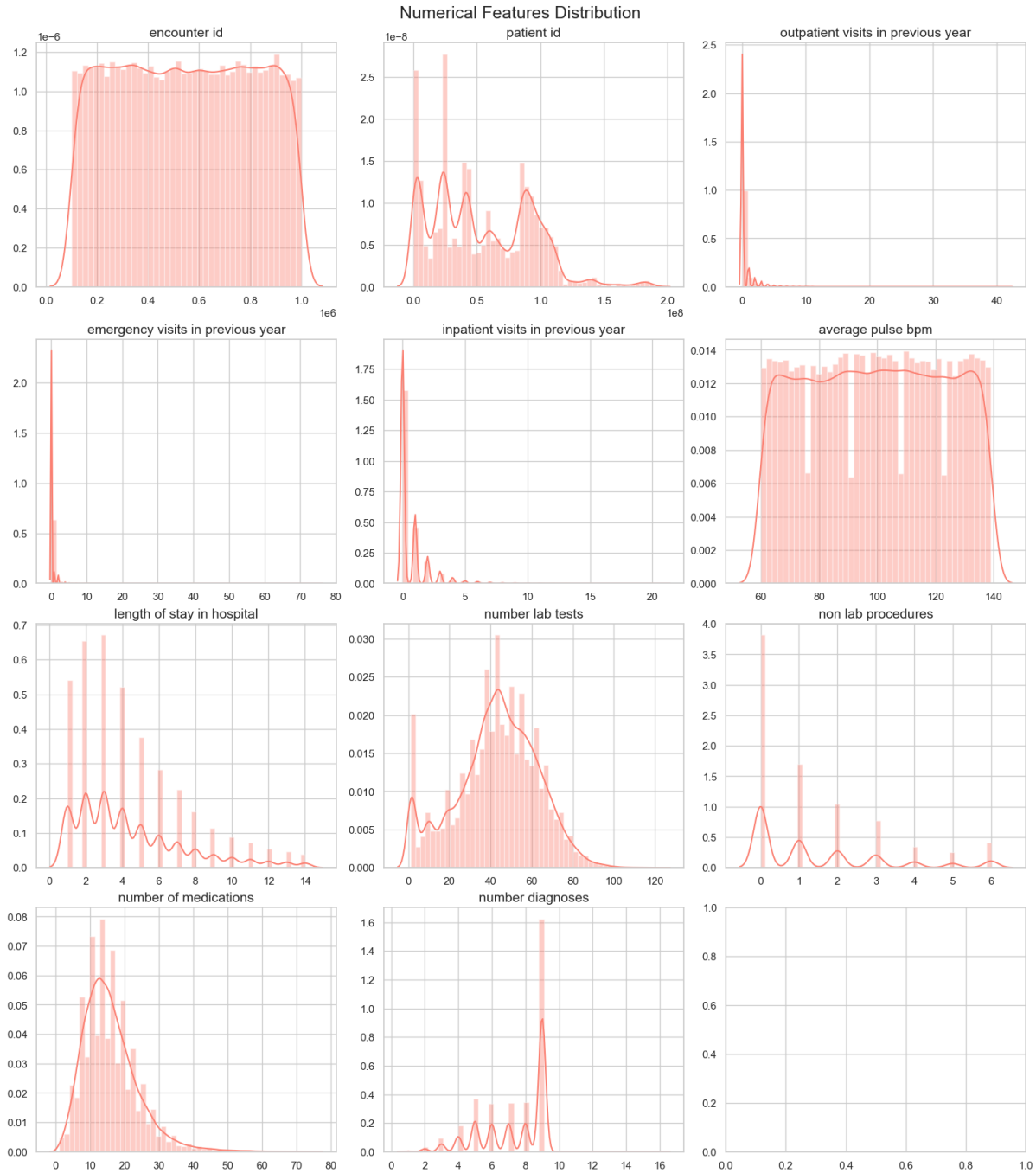


*Figure 3. Numerical Feature Distribution*

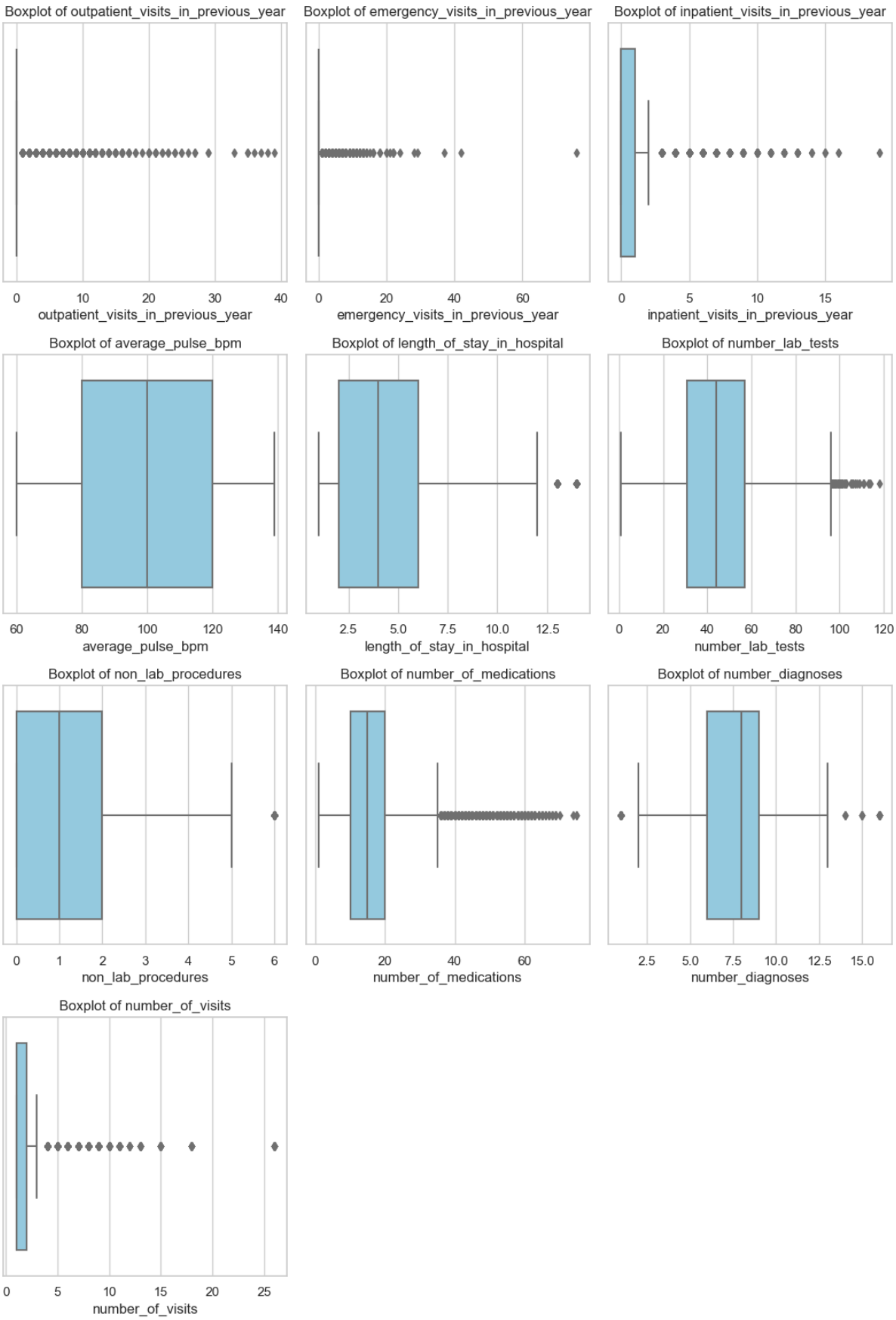


*Figure 4. Visual Outlier Detection*

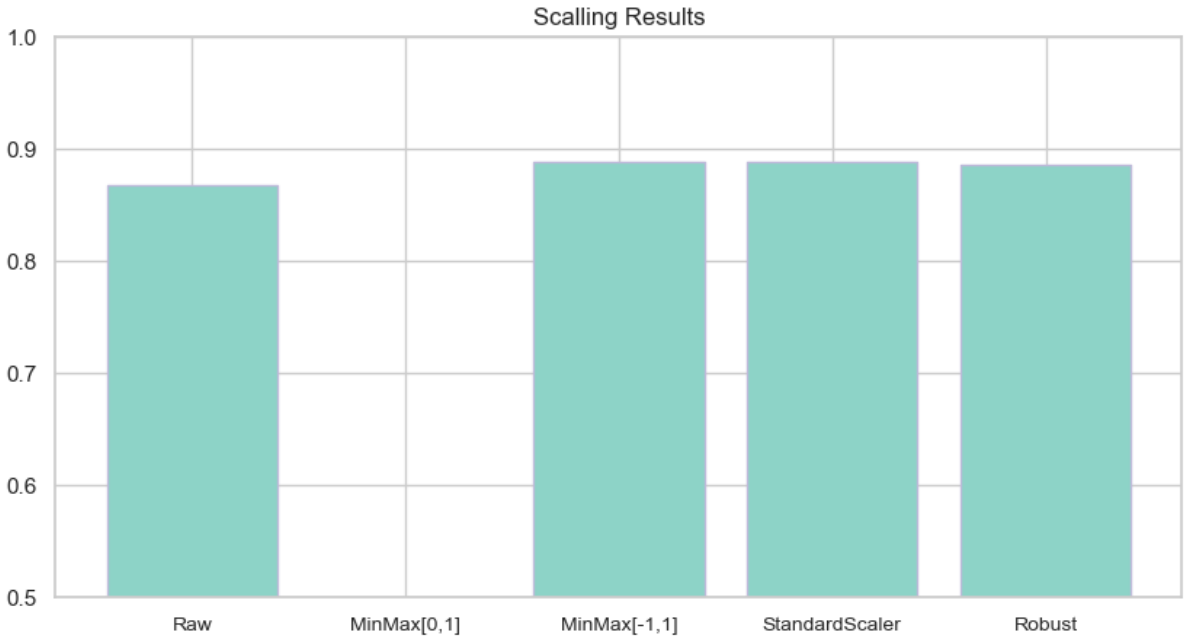


*Figure 5. Scalling for Binary Classification*

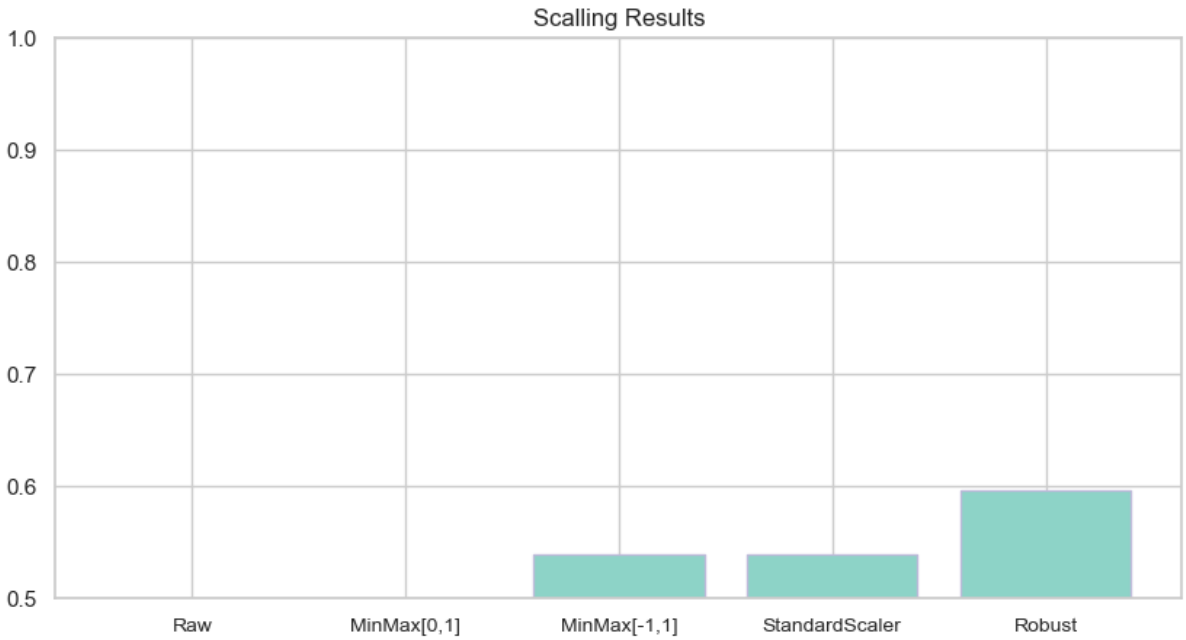


*Figure 6. Scalling for Multiclass Classification*

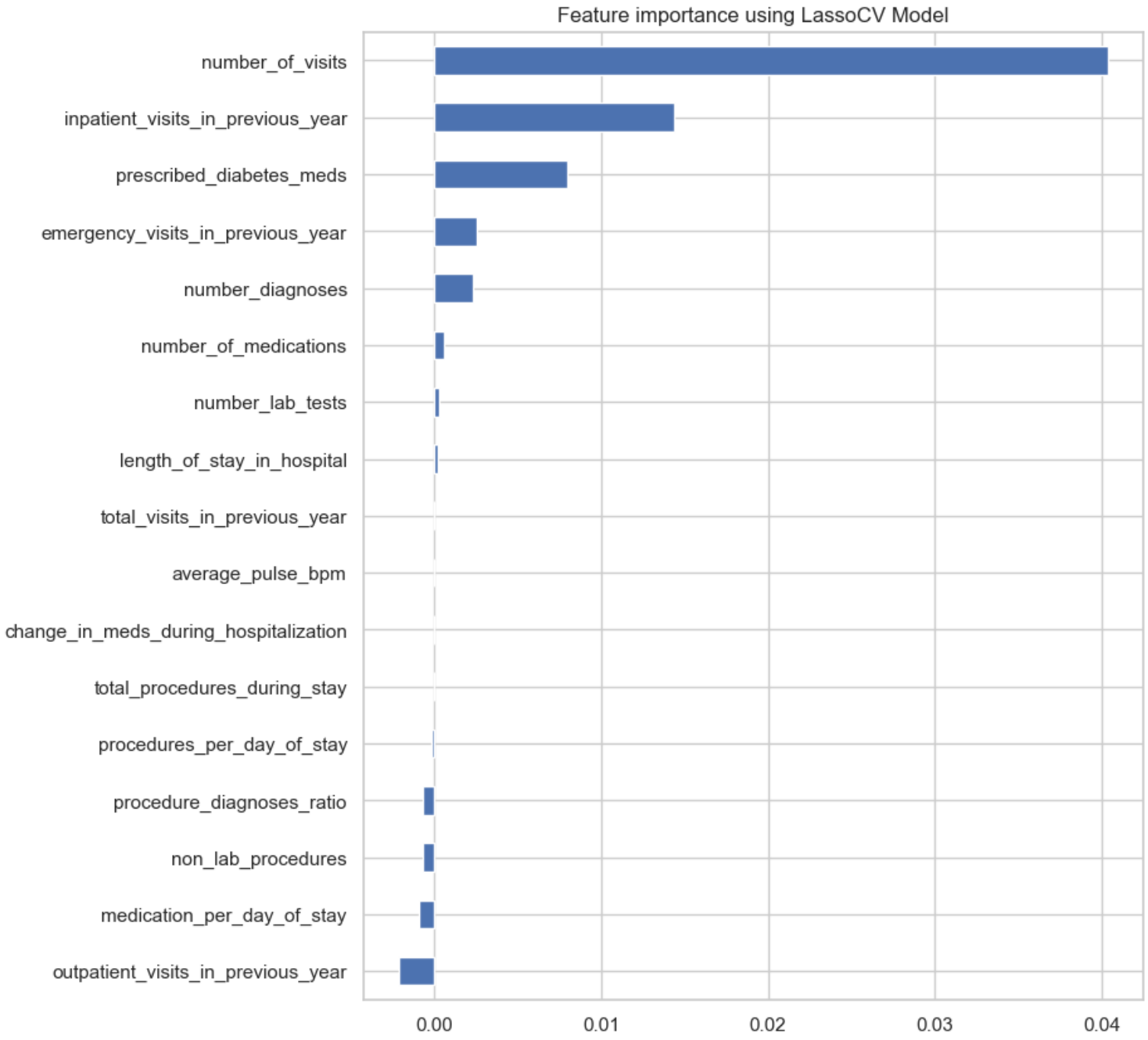


*Figure 7. Lasso Feature Selection*

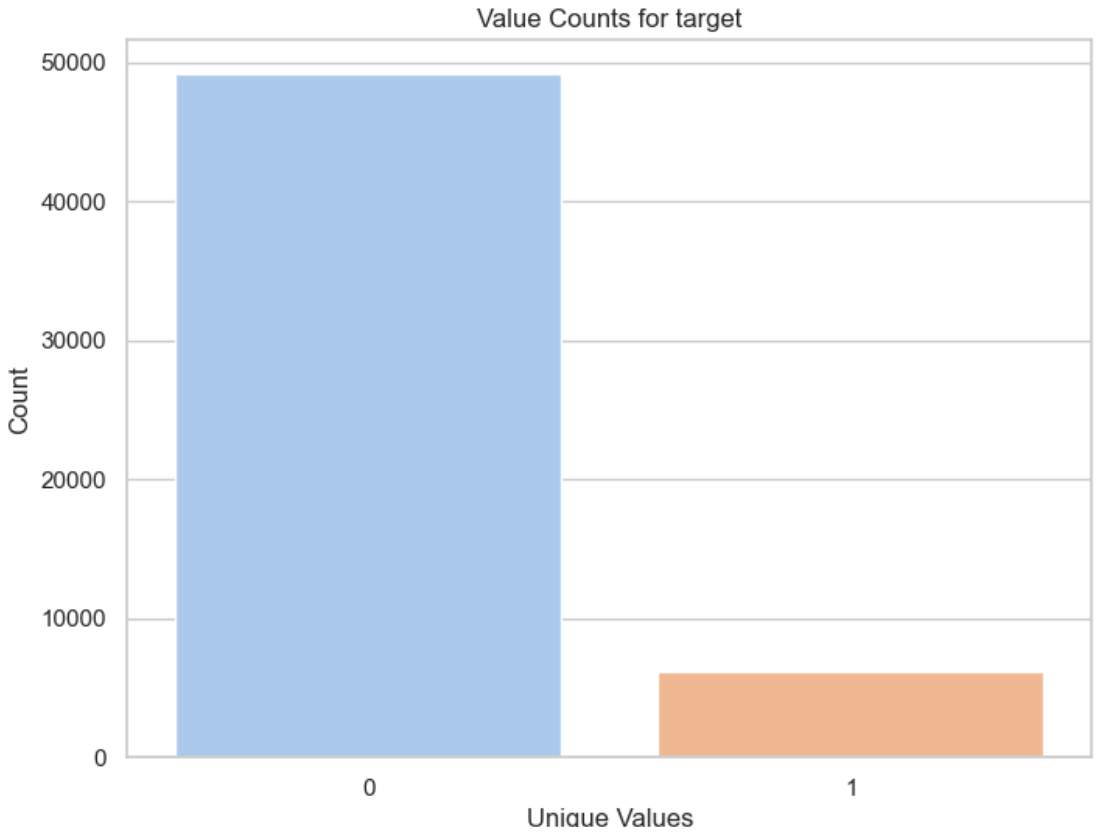


*Figure 8. Imbalanced Binary Target*


*Figure 9. Balanced Binary Target*

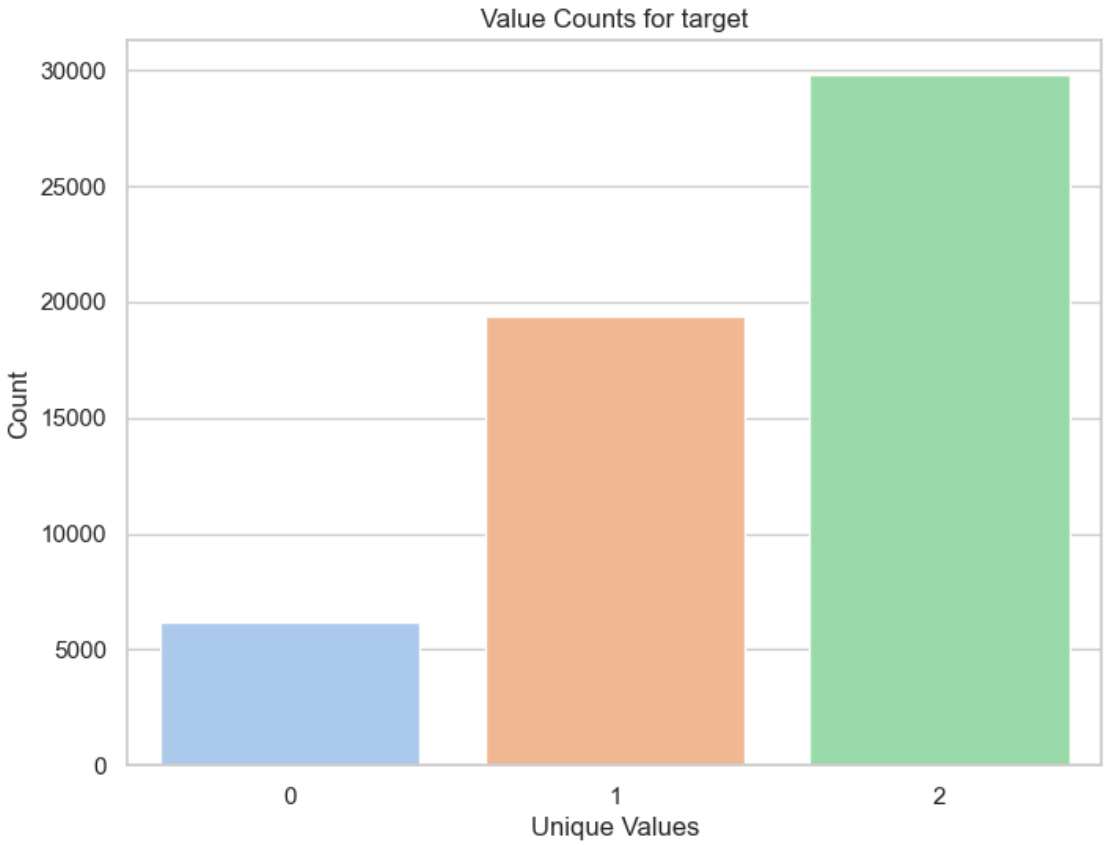


*Figure 10. Imbalanced Multiclass Target*

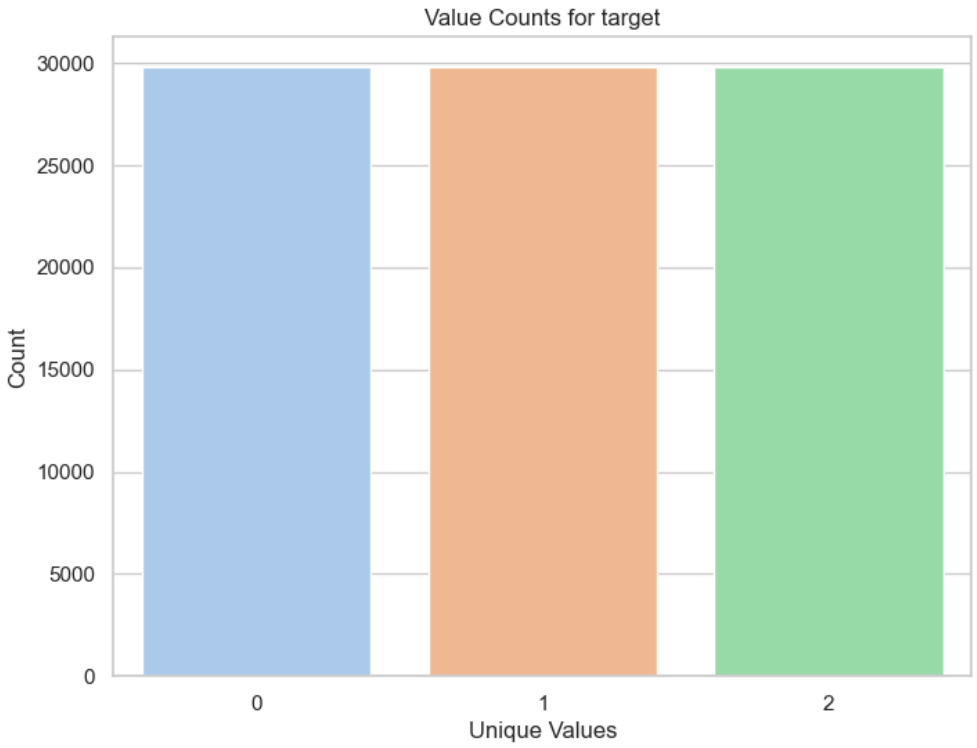


*Figure 11. Balanced Multiclass Target*