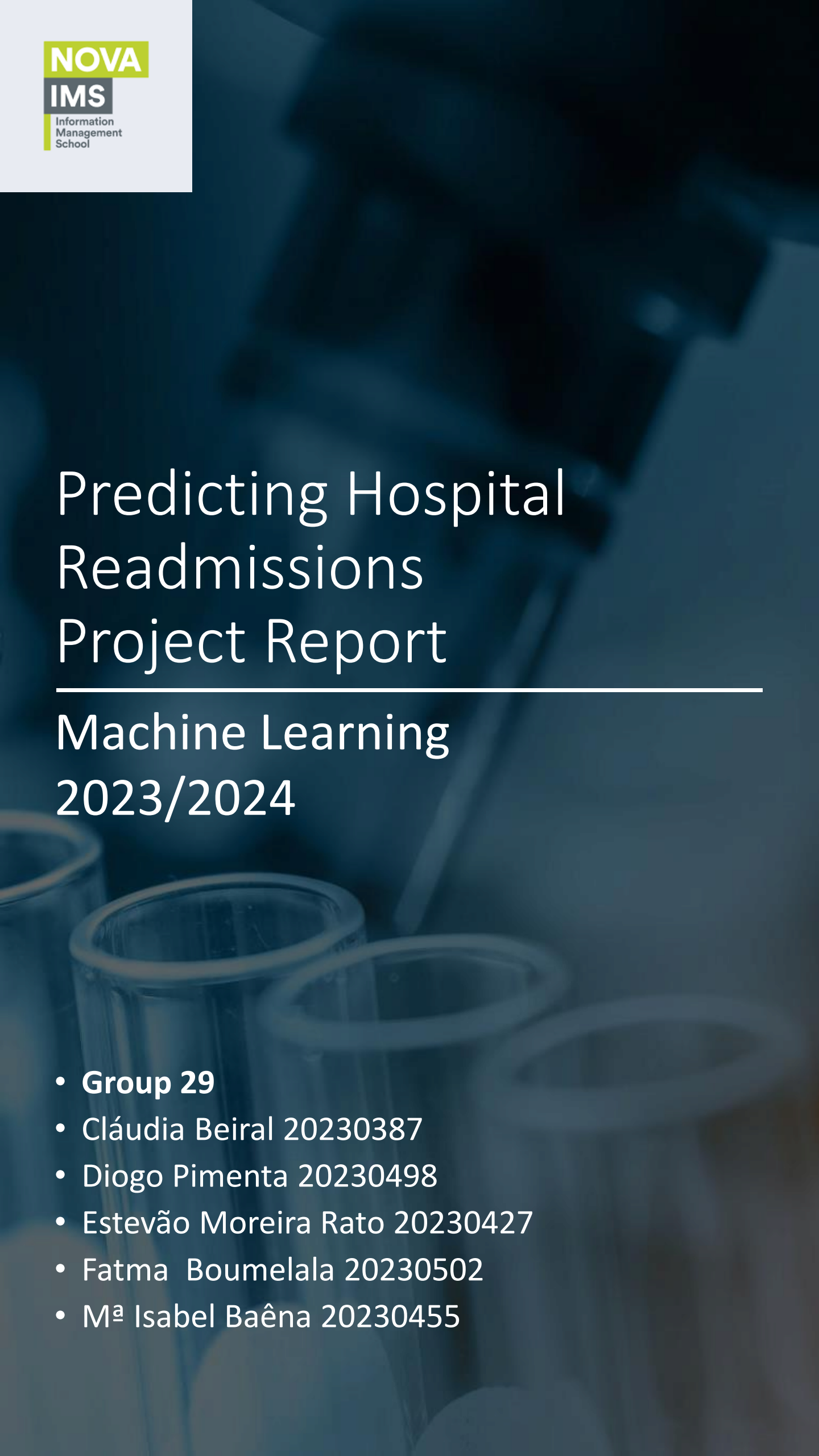

Predicting Hospital Readmissions Project Report

Machine Learning 2023/2024

- **Group 29**
- Cláudia Beiral 20230387
- Diogo Pimenta 20230498
- Estevão Moreira Rato 20230427
- Fatma Boumelala 20230502
- Mª Isabel Baêna 20230455

Abstract

In the context of healthcare quality assessment, this project addresses the critical issue of hospital readmissions for diabetic patients. Our primary goal is to create two predictive models that can anticipate when diabetic patients might be readmitted to a hospital and, when appliable, the timeframe of the readmission. Although false positives are a concern, our secondary goal focuses on minimizing false negatives, which represents the patients who were wrongly predicted as not being readmitted.

To achieve this, a methodology which involved a thorough exploration and preprocessing of the data was followed. We attended to data incoherences and imbalances in the dataset, transformed features and performed feature selection to optimize the performance of the predictive models. Several machine learning models were developed and tested, including ensemble methods. Random Forest turned out to be the best performing algorithm for both the binary multiclass classifications with a f1 score of 0.32 and 0.65, respectively.

In conclusion, this project successfully developed decently accurate models with satisfactory recall for anticipating diabetic patient readmissions.

I Introduction

Hospital readmissions are considered a reliable indicator of the effectiveness and quality of the healthcare provided to patients. The ability to predict such readmissions becomes a vital tool for enhancing patient care and targeting cost reductions.

The goal of our project is on the development of a predictive model tailored to address the issue of hospital readmissions of diabetic patients. It includes two classification models: a binary classification and a multiclass classification. The binary seeks to accurately predict whether a patient will be readmitted to the hospital within 30 days of discharge. The multiclass predicts the timeframe of a patient's readmission.

The project involves comprehensive data preprocessing to prepare the data for the predictive models followed by the development and optimization of several machine learning models to tackle the proposed task of forecasting hospital readmissions.

II Data exploration and preprocessing

Data description

The dataset provided consists of 71236 medical records for 53985 American patients. Each instance regards a hospital admission in which any kind of diabetes was entered to the system as a diagnosis.

Medical records in the dataset include 31 attributes including the 2 target variables that indicate the patient's readmission status and, when applicable, the timeframe in which the readmission occurred which is the multiclass target.

The patient attributes encompass demographics, healthcare utilization, clinical indicators and medication details providing valuable insights for analysis and predictive modelling.

Data incoherence and Missing values

In our initial data exploration, we checked for duplicates, recognizing their potential to lead to overfitting during model training. However, our thorough examination revealed no instances of duplicated records, ensuring a clean starting point for further analysis. Nevertheless, we did find some data inconsistencies, in “admission_type” there are newborns older than 5 years old, which is not realistic, so we decided to drop these records.

When looking at the unique values for each feature, it was noted that all patients are from the USA. As there is no variability in this column, we decided to drop it as it does not provide any patient-specific information.

Moving on to the crucial task of handling missing values, we first focused on finding the conventional “NaN” representation. Upon a closer analysis, it was observed that the missing values were represented with strings like: “?”, “Not mapped”, “Not available” or “Unknown/Invalid”. This process exposed a significant presence of missing data throughout the dataset, as shown by Fig.1, and the challenge arose in deciding how to handle/replace its the different formats.

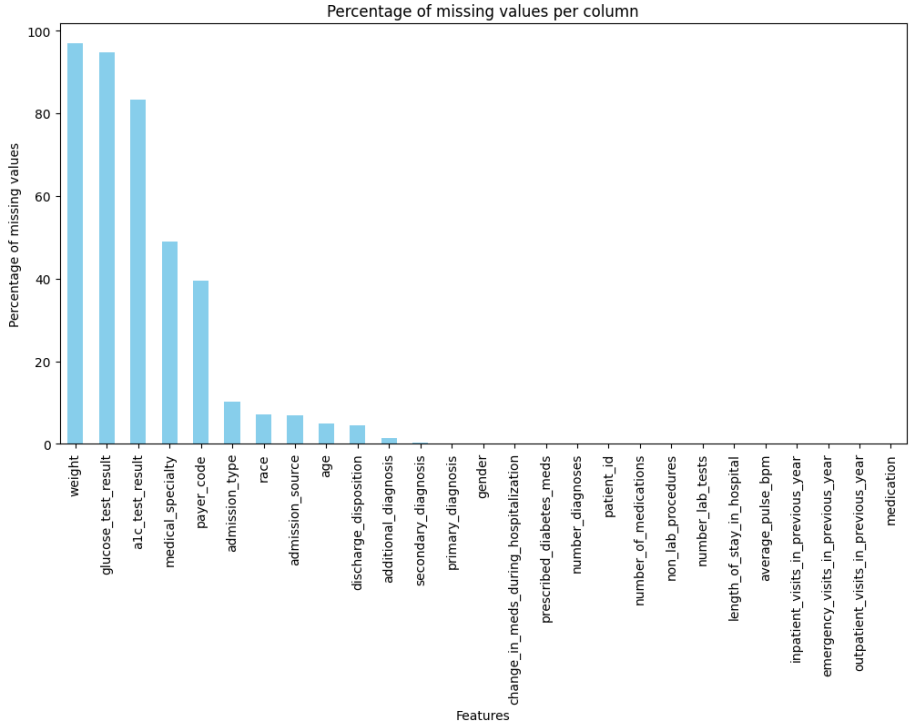

Fig.1. Percentage of missing values per column

In addition, we opted to drop some features that have a big percentage of missing values (higher than 30%). The ‘weight' feature, for instance, has missing values exceeding 90%, indicating a significant data gap. Similarly, the 'medical specialty' feature shows a missing value rate surpassing 40%.

In the context of test-related features ('glucose_test_result' and 'a1c_test_result'), we found out that missing values corresponded to cases where the test was not taken. In response, these missing values were replaced with a new category, 'Not_tested'.

Even though the missing values in 'payer_code' were around 40%, those were interpreted as instances where patients did not have insurance. This feature was transformed into a binary variable distinguishing between patients with insurance (coded as 1) and those without (coded as 0).

For the 'age' column, the first step was to put the values in numerical form which was archived by computing the mean for each age bracket. Then, we employed imputation using the mean calculated from the available values on the training set, associated with the same patient ID. For the 'race' column, the same strategy was employed, using the mode of each patient's race to apply imputation to handle the missing data.

For the remaining missing values, we opted to use a simple imputer, utilizing the mean for numerical variables and the mode for categorical variables. We also attempted to use KNN Imputer for numerical features, but the resulting score was inferior.

Imbalanced target

Another noteworthy observation was the imbalance in our target variable 'readmitted_binary'. A substantial majority of instances, 63,286, belonged to the 'No' (0) category while only 7950 belong to the 'Yes' (1). This imbalance posed a challenge, potentially resulting in poor model performance for the minority class.

To address this, we considered the approach of proportionately sampling both classes. Our first approach was to use the SMOTE technique which creates some artificial instances for the minority class to achieve a balanced proportion with the majority class. However, this resulted in the model overfitting, so we decided not to employ it.

We then opted to employ stratified sampling using stratifiedkfold (instead of a simply cross validation), a technique that ensures the preservation of class distribution in the sampled data. For the same reason, we decided to use the parameter class_weight = 'balanced'. The handling of imbalanced data sets the stage for more effective model training and prediction. (Fig. 2, Fig.3)

At this point, we decided to split the features into numerical and categorical for future handling.

This distinction allowed us to tailor our preprocessing steps to the nature of each feature type, ensuring a targeted and efficient approach to data transformation. The specific details of these transformations and the reasoning behind feature selection will be explained in the following section dedicated to data transformation.

Feature Engineering

- Medication:

One of the steps we took in preparing the dataset for analysis was handling the column that represents the prescribed medications for each patient. This column had a lot of unique

values, because there were a lot of different combinations between all the medications. Following the separation of each individual medications, we used one hot encoding which created a binary column for each value, assigning 1 if the medication was prescribed or 0 if not. This gives us a clear representation of the presence or absence of each medication for each row on the data set, what made it easier to analyze. We considered doing a dummy variable to clearly represent the cases in which no medications were prescribed, however this was covered by an existing feature ("prescribed_diabetes_meds"). This proved to be quite useful in the training of the model, instead of just dropping the feature because the model was probably able to find a relationship between the prescribed medications and the targeted outcomes.

- Diagnosis columns:

The columns regarding diagnosis were also comprised of many unique values.
First, we decided to group the diagnosis into a category of the list of codes for the International Statistical Classification of Diseases and Related Health Problems[1]:

- Infectious and parasitic diseases, grouped from the number 1 to 139;
- Neoplasms, grouped from the number 140 to 239;
- Diabetes, endocrine and metabolic disorders, grouped from the number 240 to 279;
- Other diseases or conditions, grouped from the number 280 to 289;
- Mental disorders, grouped from the number 290 to 319;
- Other diseases or conditions, grouped from the number 320 to 389;
- Circulatory system diseases, grouped from the number 390 to 459;
- Respiratory system diseases, grouped from the number 460 to 519;
- Digestive system diseases, grouped from the number 520 to 579;
- Genitourinary system diseases, grouped from the number 580 to 629;
- Other diseases or conditions, grouped from the number 640 to 679;
- Skin diseases, grouped from the number 680 to 709;
- Musculoskeletal disorders, grouped from the number 710 to 739;
- Other diseases or conditions, grouped from the number 740 to 759;
- Perinatal conditions, grouped from the number 760 to 779;
- Uncertain conditions, grouped from the number 780 to 799;
- Injury and poisoning, grouped from the number 800 to 999;

After, each categorization was turned into a binary feature, so that we could later figure out if a specific type of condition would have a big impact on the patient being readmitted or not. It was also considered the option of further grouping the variables, frequency encoding them in each diagnosis thus reducing the total number of variables, but later we saw that it didn't have an impact on the performance of the model.
There were some diseases with really low value counts which we decided to group into the 'Other diseases and conditions' category as an attempt to reduce dimensions. At the time of making the report we realized that it would probably have a positive impact in the performance of the model, in the sense that it could make some predictions based on those specific diagnosis.

- Admission type:

We had some doubts about the label encoding this feature because we were uncertain about the relationship between each one of the possible admission types. After some

research we found that these are indeed associated with a level of severeness resulting in the following labels:1. Emergency; 2. Urgent; 3. Elective; 4. Newborn; 5. Trauma[2].

This allowed for a more efficient utilization of this variable, because there really is an ordinal relationship between the categories in a hierarchical sense, thus if we were to encode this variable in some other way, we would probably lose a bit of performance on the model.

- Admission source:

In this variable we decided to group all the transfers from other hospitals independently of its source into just one category which is 'Transfer from Another Health Facility', this way we had less features to deal with in our data.

- One hot encoding:

The features that were selected to be one-hot-encoded were the race, the 'discharge disposition', the 'glucose test result', the 'a1c test result', and 'admission source'. This leads to getting a feature created for each unique category for all the features in these selected columns.

- Binary features:

There are some features that possess only two possible outcomes. These features are 'gender', 'change in medication during hospitalization' and 'prescribed diabetes medications'. To make the model training possible, binary categorical features need to be transformed into dummy variables, giving them a numerical representation instead of 'yes' or 'no', or 'male' and 'female' in the case of the gender, which some models cannot handle. Since our target variable was also binary, we applied the same principle to it, transforming 'No' into 0, and 'Yes' into 1.

- Creating new feature:

Lastly, we created some features that could be important for the model. The first one is 'hospital visits', which counts the number of times each 'patient ID' has been to the hospital in the dataset. This variable proved to be of much value, as we observed it had a significant impact on the improvement of the score, also being considered the most important feature in the feature selection as we will discuss later.

To further improve the model, we also added 'n_medications/lenght_of_stay' which indicates the number of medications administrated to a patient during the encounter divided by the number of days a patient has been in the hospital which indicates the average of medications administered by day.

The last feature created was 'Emergency_visits/Total_visits', which provides the proportion of emergency visits in relation to the total number of visits (inpatient, outpatient, and emergency visits in previous year). In the instances where the patient did not go to the hospital in the previous year, we set the value of the feature to be 0. This feature also proved to be somewhat important as we will later review the feature selection techniques we used.

- Outliers & Scaling Data:

Outliers are data points that deviate significantly from most observations in a dataset. These values need to be assessed as they have the potential to negatively impact on the model's performance.

Our first step was to employ boxplots to identify outliers for each non-binary numerical feature in our dataset. The boxplots show outliers in all the plotted features except for "average_pulse_bpm" (Fig.4).

Our first approach was to perform outlier removal using the interquartile range. We opted not to select it as our method because it resulted in the deletion of a significant portion (around 40%) of the instances of the training dataset.

Subsequently, we applied feature-specific filters to exclude data points exceeding certain thresholds. The thresholds were defined empirically by looking at the values outside the whiskers of each boxplot. We concluded that due to the subjectivity of this method, this would not be the most reliable approach.

Furthermore, considering the context of the outliers within each feature and the nature of our dataset, we did not consider it a good option to drop these values.

Winsorizing, which consists of transforming the values of the outliers to be less extreme [3], was also considered. However, this approach also requires choosing the limit percentiles which may lead to some error since it's arbitrary.

Hence, our approach is to keep these observations while minimizing their potential influence on the models. We applied this strategy by employing a robust scaler to temper the impact of outliers on the modelling process[4]. The robust scaler subtracts the median to center the data and scales the data according to the IQR.

The use of robust statistics diminishes the impact of outliers, as these measures are less sensitive to extreme values.

During this step, we also address the need to scale the features, preparing them for model training.

Feature Selection

The feature selection consisted in 5 steps:

1. Check the variance of each feature:

If a feature is constant, it does not contribute to our model because it does not help distinguish between different classes. Essentially, it becomes redundant and irrelevant training the model. However, at this point we did not have any invariant features, so we did not drop anything.

2. See the correlation between features:

It is important to avoid having features that share similar information in the model. If we have correlated features, this means we have redundancy, because two highly correlated features explain basically the same things. This is not good for training our model,

redundancy does not add useful information and can complicate the learning process. To analyze the correlation, we plotted a heat map with all correlations above the absolute value of 0.8 (Fig.5).

We observed that 'Emergency_visits/total_visits' and 'Emergency_visits' are 100% correlated. Therefore, only one of them should be included in the model. However, we haven't discarded any of them until we know the decision from other feature selection methods.

3. Feature Importance with random forest

Since random forest has a tool that helps us do feature selection, we thought it would be good to take advantage of it.

After analyzing feature importance (Fig. 6), we did a cumulative importance plot (Fig.7) to decide on the number of features to retain. This plot showed that the cumulative importance stabilizes around 80 features. Beyond this point, additional features contribute minimally to increasing its overall importance and will only add dimensions, potentially leading to the curse of dimensionality.

4. RFECV with random forest

RFE (recursive feature elimination) is a method that trains a model, identifies the least important feature, and removes it. RFECV combines this with cross-validation, repeating the process and adjusting the model based on the F1 score. The aim is to find the best set of features for optimal model performance. The RFECV returned that the best number of features is 2 (Fig.8).

5. Lasso

We decided to use Lasso as a feature selection method because when we train lasso, we can see the coefficients attributed to each feature with the ones having coefficient = 0 being considered ‘not important’ for lasso. This allows us to assess their importance and make decisions on which features to keep or discard (Fig.9).

For the final decision we chose the features that have been selected by at least 2 methods (Feature importance, RFECV and Lasso). We kept 51 features.

III Binary Classification

The binary classification model requires a numerical target variable, a task that was addressed during the preprocessing stage.

To test the models, we opted for cross-validation (stratified k fold cross validation to account for the imbalanced classes) instead of the holdout method. In cross-validation, every instance in the training set is used at least once to train the model. The model is trained with k different subsets (we used 5). Using the holdout method would result in fewer instances, potentially negatively impacting the learning of the model.

After testing some models with the selected features, we verified our scores were not satisfactory. We figured it would be logical for our models to learn from the frequency of patient readmissions. For example, we reasoned that a patient who has been readmitted multiple times might have a higher likelihood of being readmitted again. For this purpose, we

created the feature hospital visits which, as mentioned before, were revealed to be essential to improve the overall performance of the models.

To further improve our models, and tailoring them to our dataset, we need to account for hyperparameter optimization. For each model tested we attempted to perform a grid search, which searches through a predefined set of hyperparameter values to evaluate the model's performance for each combination of hyperparameters. However, due to computational constraints, the exhaustive nature of grid search was not viable.

We opted to use random search which is acclaimed for discovering highly effective parameter combinations in significantly less time compared to grid search[1]. It consists of randomly sampling combinations of hyperparameter values from a predefined search space, assessing the model's performance and choosing the hyperparameter configuration that results in the best performance once a predefined number of iterations is completed. In this approach, the search space needs to be carefully selected to ensure that the best combinations are not overlooked.

To evaluate our models, we used f1_score not only because it is the Kaggle metric for our project bur also because f1 score combines precision and recall so evaluates both false positives and false negatives which is what we need because we have imbalanced classes.

We think that accuracy is not the most reliable metric in the context of our dataset which is very imbalanced. Accuracy does not consider false positives and false negatives which may lead to overall accuracy appearing high even when the model is performing poorly in the minority class. Also, in the context of healthcare, false positives and false negatives are important to make decisions.

We also want to highlight that before performing random search for each model, it is compulsory to initialize the models with class_weight=balanced because we have imbalanced classes. Otherwise, the f1 score for all the models is always 0 (regardless of performing random search).

On our project we tried to incorporate a variety of models. Beginning with binary classification, we started with random forests because they were somewhat easy to understand and would give us a good baseline to determine if we had pre-processed our data well enough. This model has been the most consistent throughout the project, always providing the best scores. Nevertheless, we also tried using some other models, however we did have some difficulties implementing random search for some of them because the model took too long to run either from the number of individuals the dataset has or might have been because of the high dimensionality we got from the preprocessing. So, we went with the default parameters for the ones we couldn't perform a random search (Neural Networks, SVM, bagging with Logistic Regression and adaboost with logistic regression).

The following are the models we used: Decision Trees, Logistic Regression, KNN, Naive Bayes, Neural Networks, SVM and a variety of ensemble models, one being the one previously mentioned, the random forest. The remaining are bagging decision tree, bagging logistic regression, bagging naive bayes, adaboost decision tree, adaboost logistic regression, adaboost naive bayes and finally gradient boosting.

Ensemble methods play a crucial role in improving the predictive performance of models. The use of various models along with ensemble techniques, creates a combined effect that's better than individual models. One major benefit of ensembles is their capacity to prevent overfitting, ensuring a more dependable and consistent prediction by combining results from multiple

learners. Bagging methods, as shown by Random Forest and Bagging Decision Trees, aim to decrease variability by training models on different parts of the dataset and then merging their predictions.

Random Forest achieved the highest F1 score among the ensemble models which was 0.33. Bagging Logistic Regression and Bagging Naive Bayes also show improved scores compared to their normal counterparts. However, Adaboost models did not outperform the normal models in this case, except in the decision tree. (Fig 10)

Although the random forest had the best f1 score, the bagging with logistic regression and SVM with default parameters were very close to the random forest. So, all three could be good choices for our final model. (Fig. 11)

Still, our chosen model was the random forest with n_estimators=100, min_samples_split=10, min_samples_leaf = 1, max_depth =10 and class_weight = 'balanced'.

IV Multiclass classification

For the multiclass classification the only additional pre-processing step was to transform the target into numerical, we transformed 'No' into 0, '<30 days' into 1 and '>30 days' into 2.

We also used random search and stratified k fold cross validation for the same reasons as before.

To evaluate the models, instead of using a binary f1 score we opted by using a weighted f1 score because we have 3 classes and the 0 appears much more times than the others and in multiclass classification, weighted F1-score is used to account for imbalances in class distribution. It gives more importance to less frequent classes, ensuring a fair representation of the model's performance. The weighted F1-score is calculated by assigning weights to each class based on their relative frequencies in the training data, providing a more balanced evaluation, especially in scenarios with class imbalances.

The models we chose for the multiclass were the same as we did in binary, we only adjusted the parameters if needed and, again we used ensembles to try to improve our model's performance (which was succeeded in all models we used with ensembles) (Fig .12)

The results were very similar, the random forest was the one that performed the best (f1_score, however gradient boosting, and neural networks could also be a good choice to be our final model because they have similar f1_scores. (Fig.13)

We chose the random forest as our final model with n_estimators=150, min_samples_split=10, min_samples_leaf=1,max_features='log2',max_depth=30, criterion='entropy', bootstrap=False.

Conclusion:

Looking at the confusion matrix in the binary classification we verified that the recall of the final model is 0.715 and the precision is 0.214. This indicates that our model has less false negatives than false positives which means the model is more likely to predict that a patient is going to be readmitted when he will not, than the other way around which is what we expected since the beginning. Given the context of our project, a false positive is preferable to a false negative thus, recall might be more critical than precision as it considers the false negatives.

In the multiclass classification the precision is 0.67 and the recall is 0.69 which means that the number of false negatives and false positives is very similar meaning the model is making positive predictions with a balanced consideration of both false positives and false negatives.

During the development of this project, we tried to deal with imbalanced classes using SMOTE however our model was overfitting, even if we adjusted the parameters so, we decided not to use. Also, in a way to address the improvement of our score, we realized we could have done some things differently:

- KNN imputer - experimenting with a larger range of k.
- Admission Source – Instead of grouping all the 'Transferred' we could have tried to group the ones that have a small number of value counts.
- For the diagnosis we were not sure if the primary diagnosis means the first one or the principal one, if it is the first one our method can be right, however if it is the second option, we ended up loosing the importance of that diagnose since we merge the 3 diagnoses into 1.

With the help of more powerful resources, we would have been able to run more powerful models and see if they outperform the ones tested on this project. With that being said, we concluded that the best model to predict hospital readmissions for this specific Dataset is the Random Forest.

References:

[1] https://en.wikipedia.org/wiki/List_of_ICD-9_codes
[2] https://resdac.org/cms-data/variables/admission-type-code
[3] https://digitalcommons.wayne.edu/cgi/viewcontent.cgi?article=1926&context=jmasm
[4](Swamynathan, M. (2017). Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python (p. 248))

Anex:

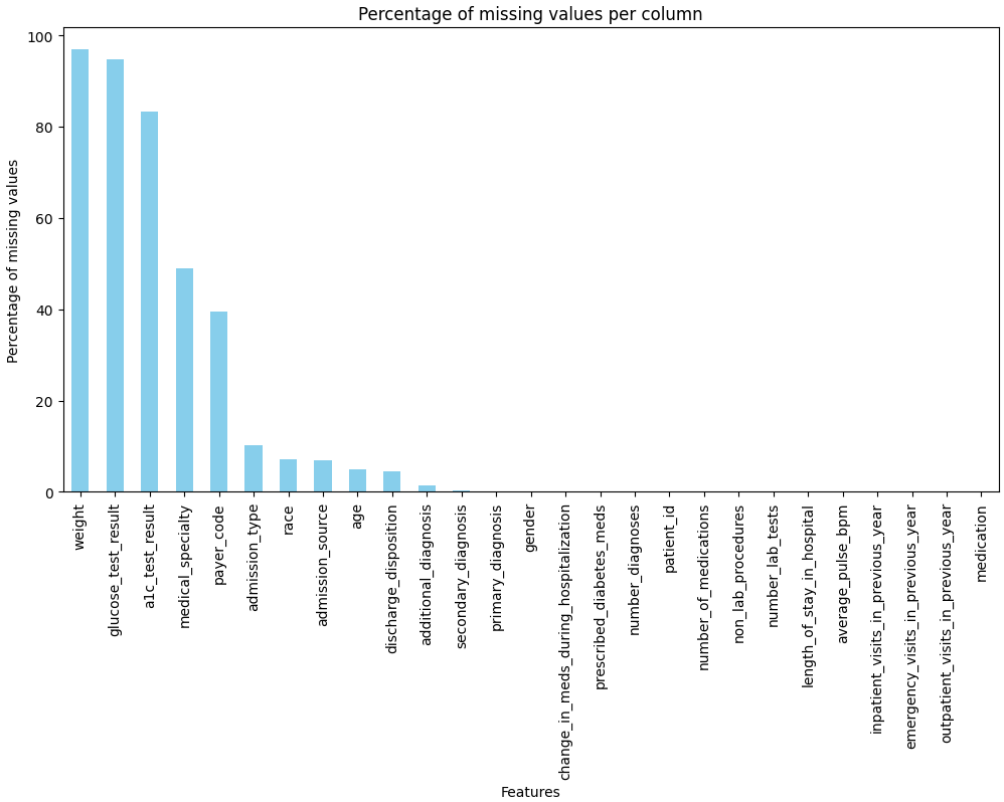

Fig 1. Percentage of missing values per column

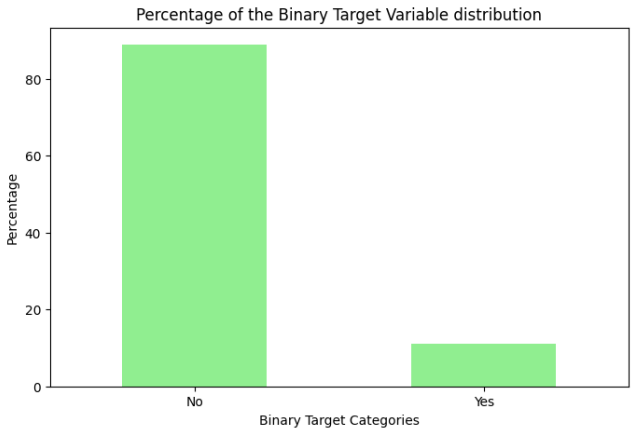

Fig 2. Binary Target Distribution

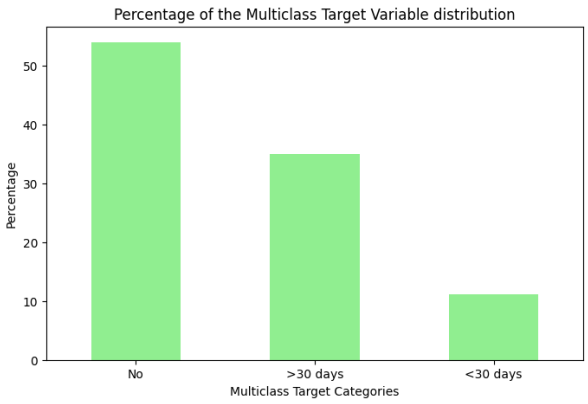

Fig3. Multiclass target distribution

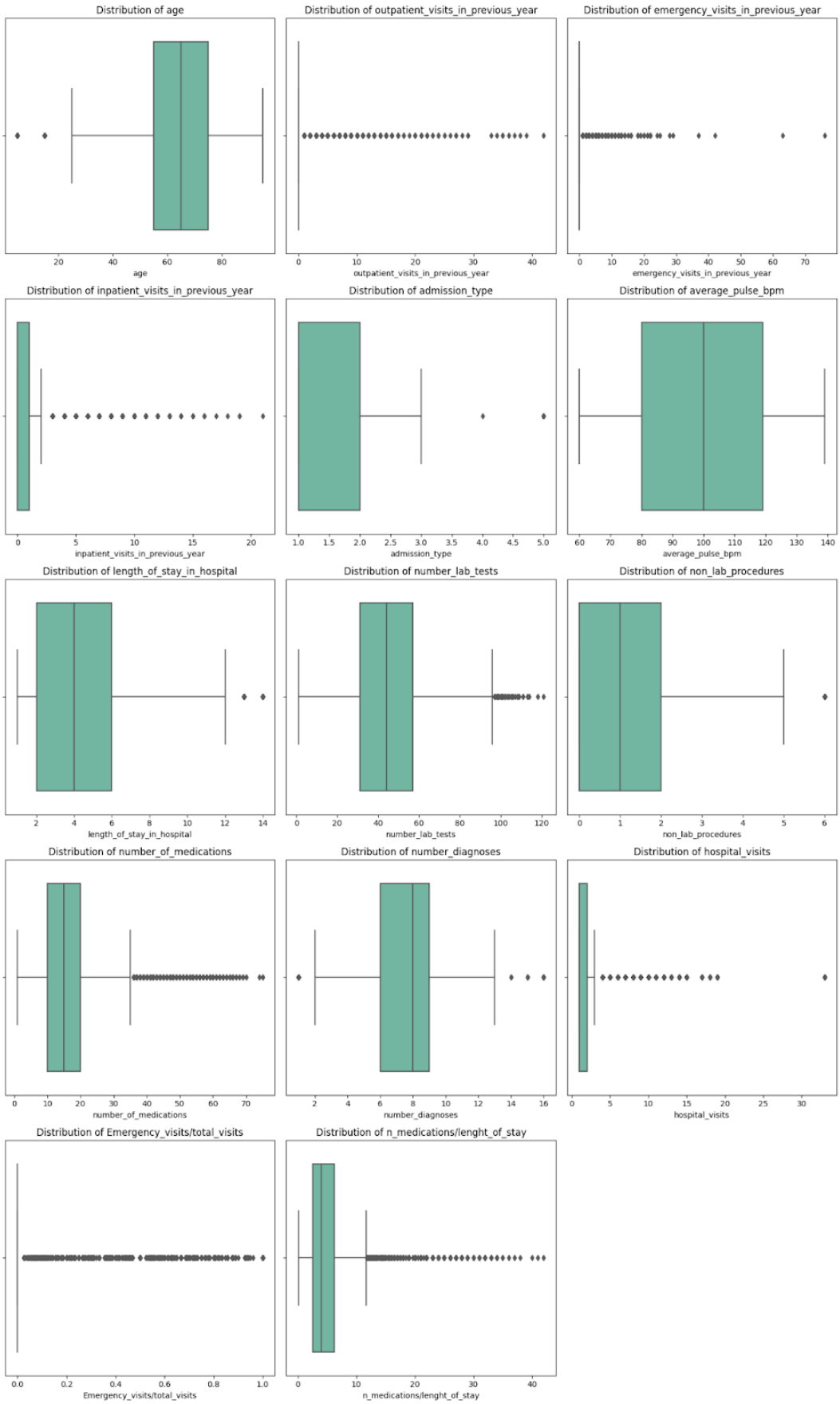

Fig 4. Boxplot of numerical features

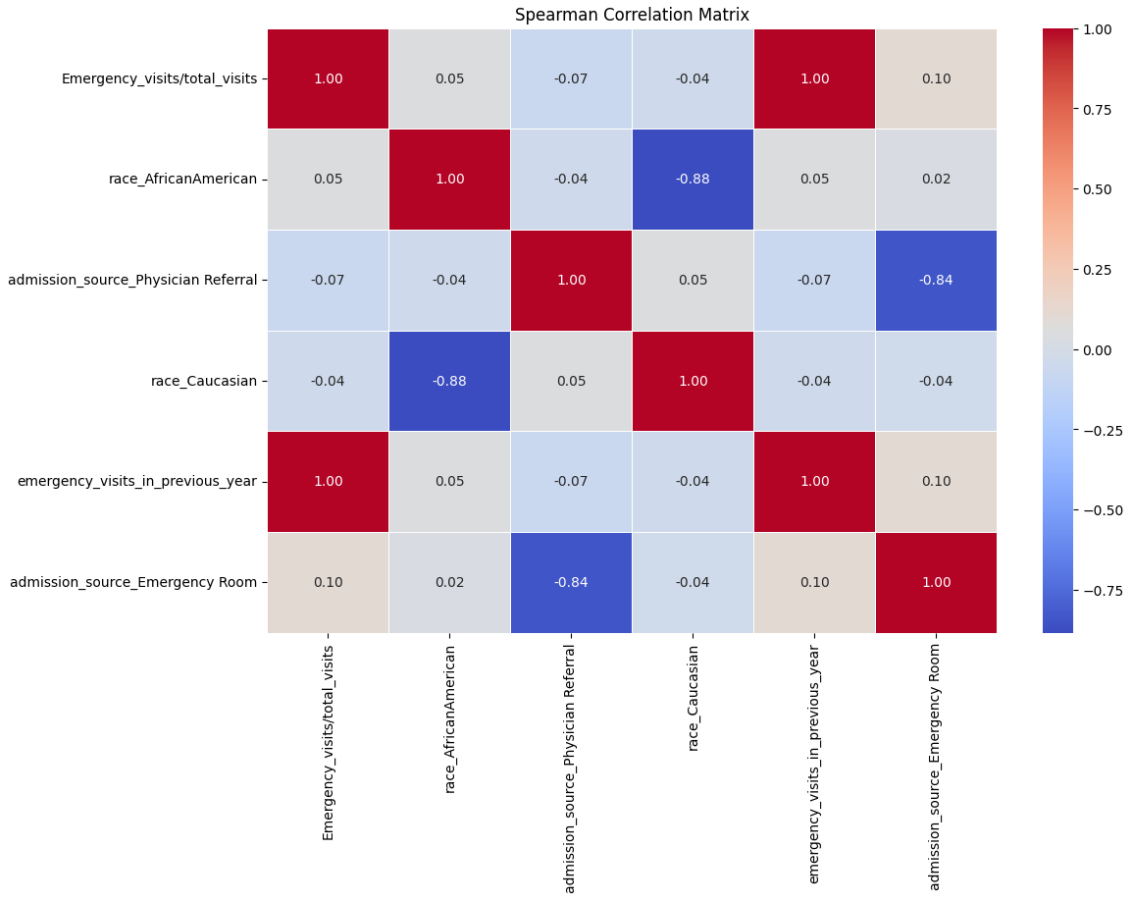

Fig 5. Correlation between features

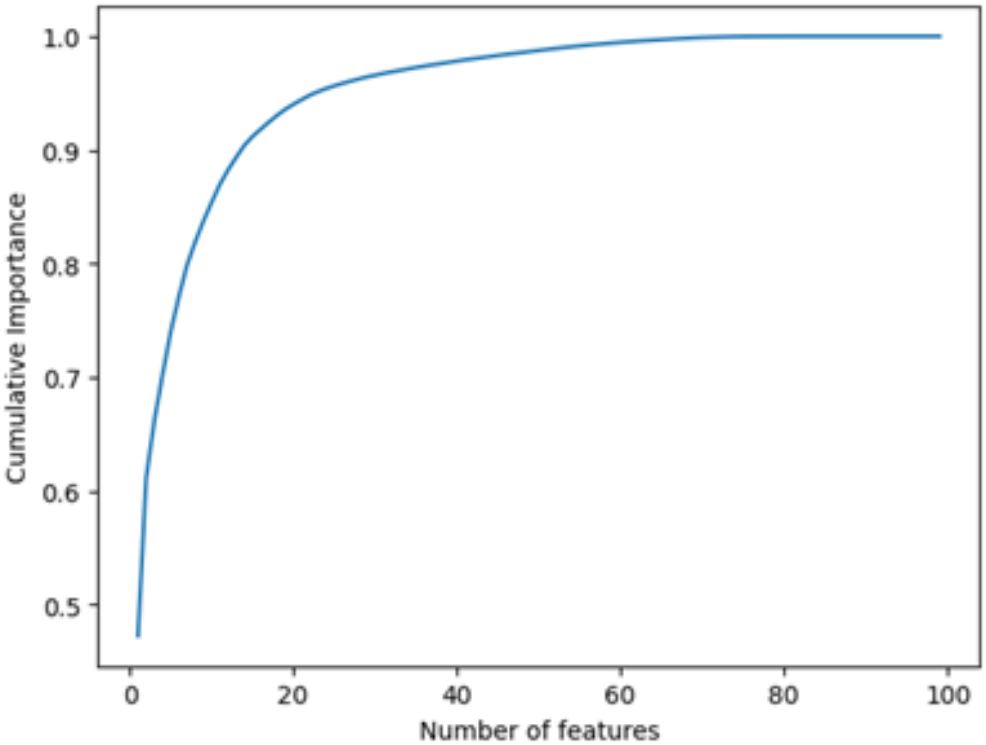

Fig 7. Cumulative Importance vs Number of features Random Forest

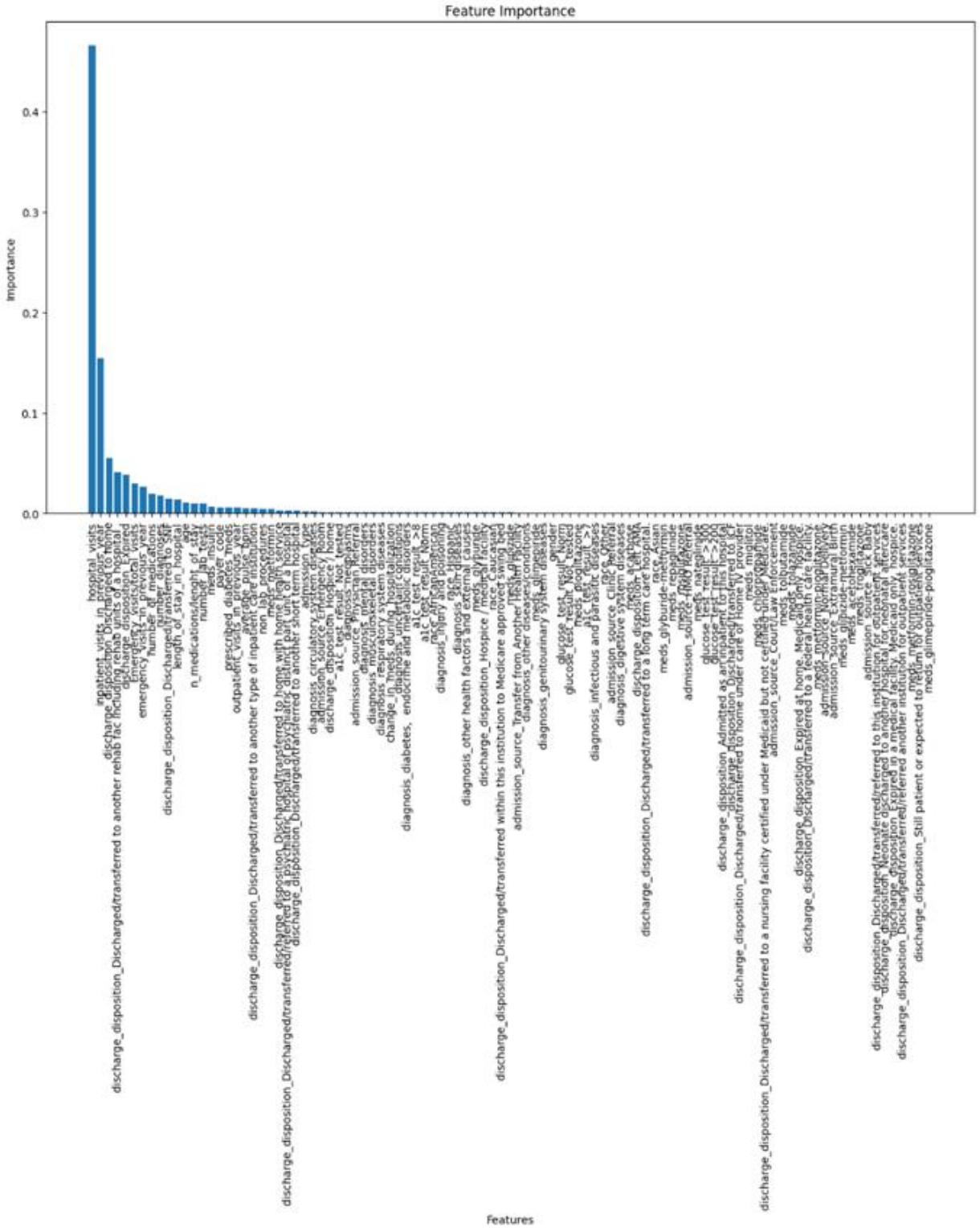

Fig 6. Feature Importance Random Forest

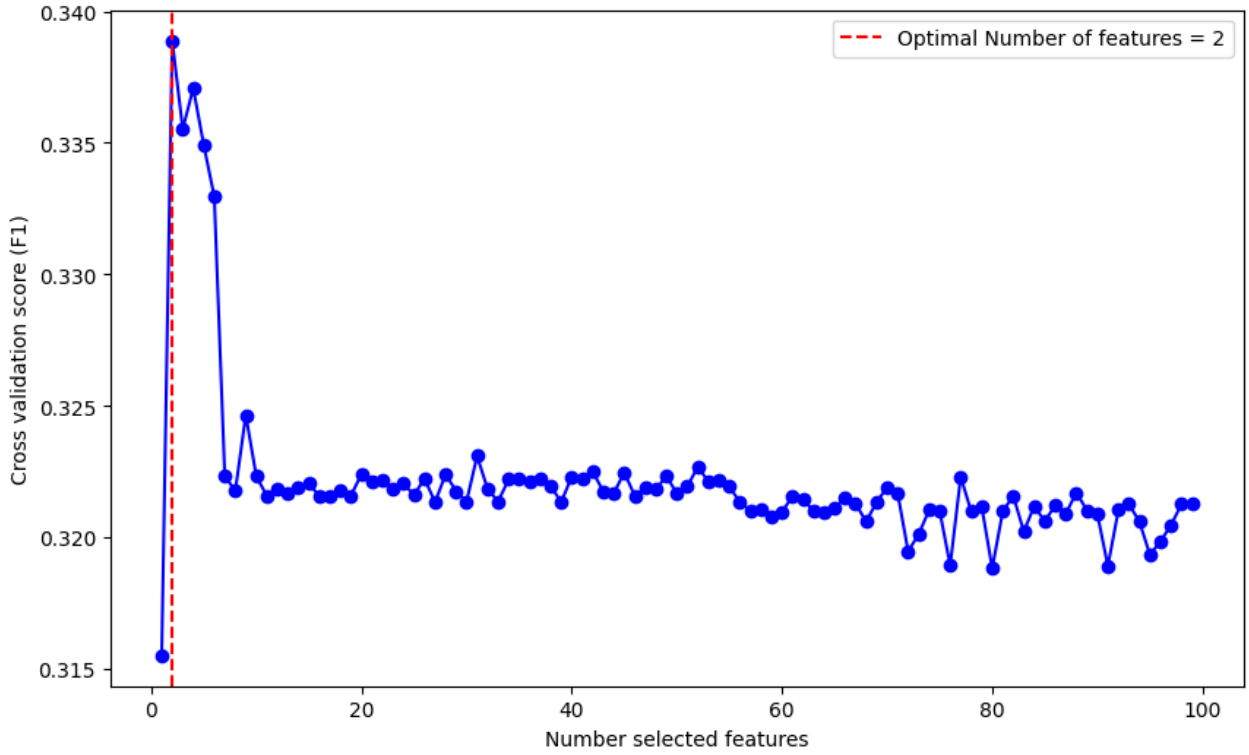

Fig 8. Best number of features chosen by RFECV

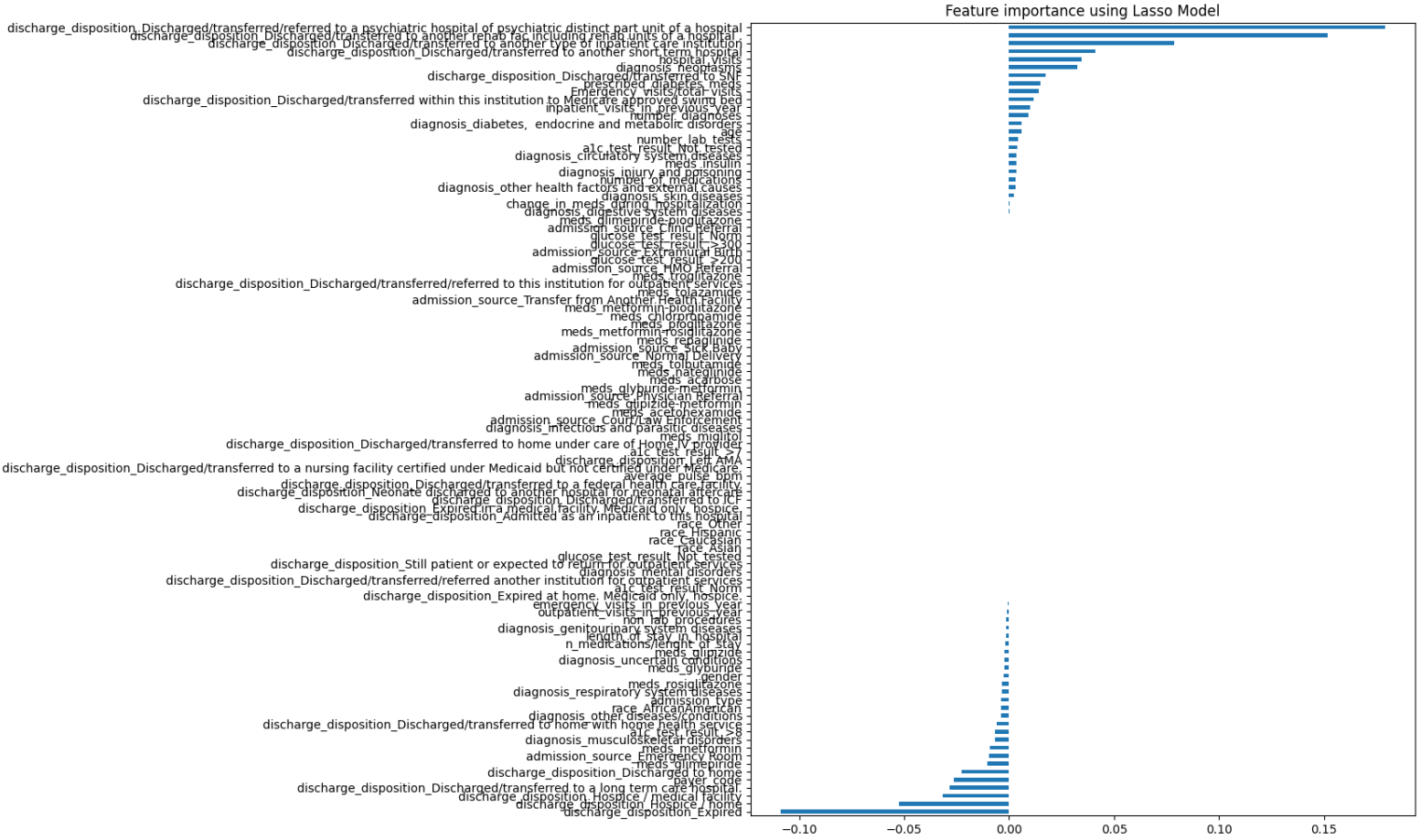

Fig 9. Lasso feature selection

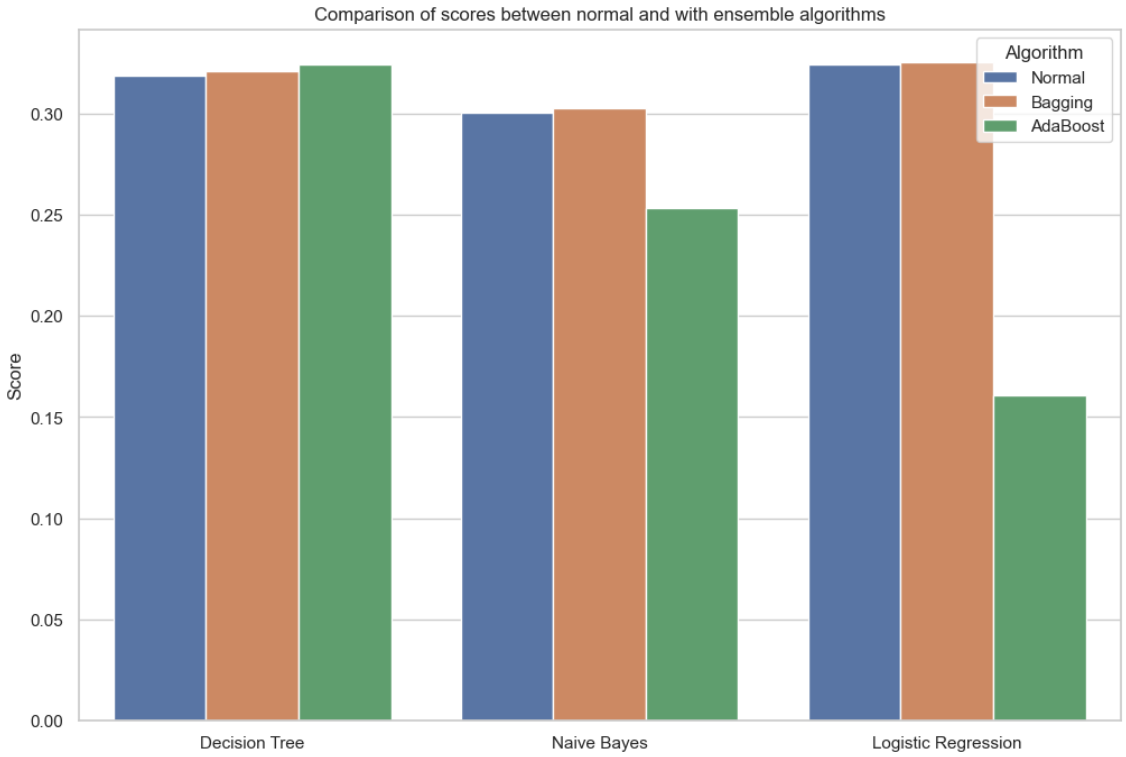

Fig 10. Normal models vs ensemble ones (binary classification)

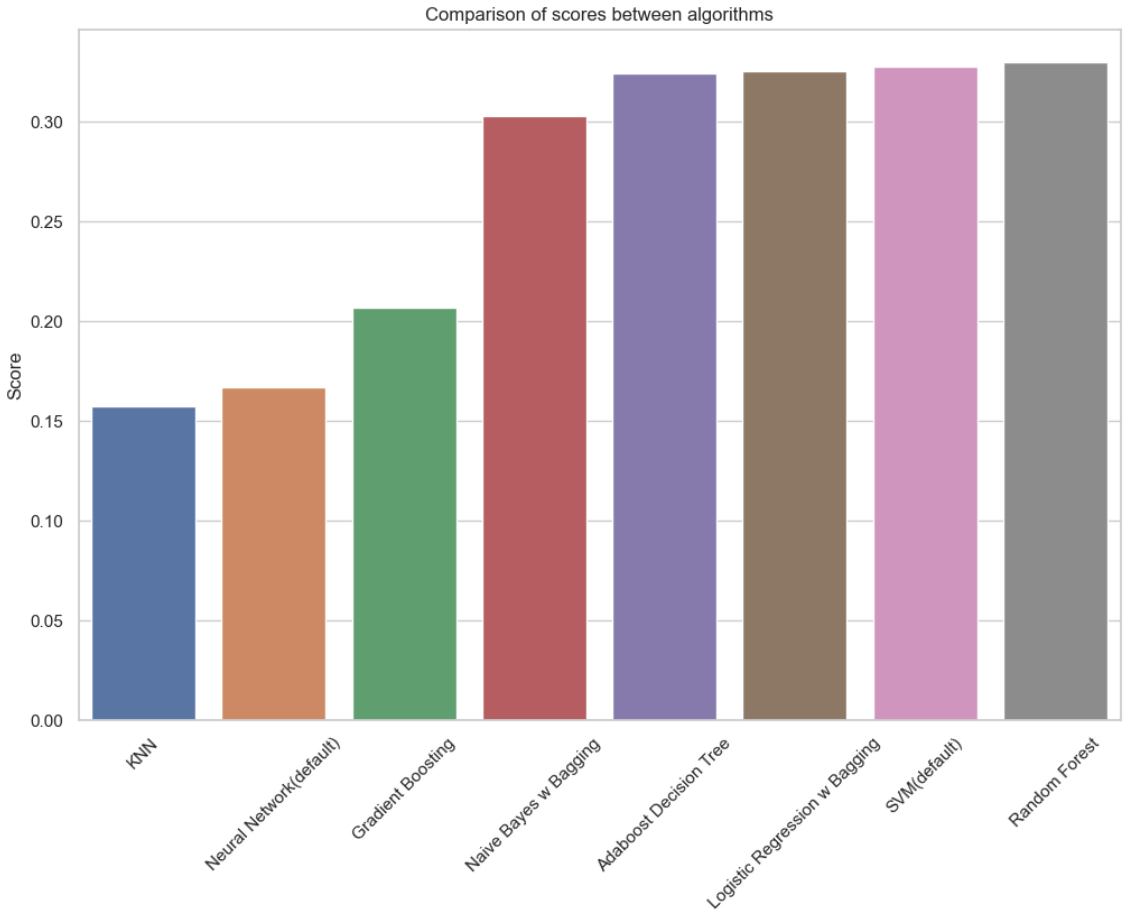

Fig. 11. Model performance (binary classification)

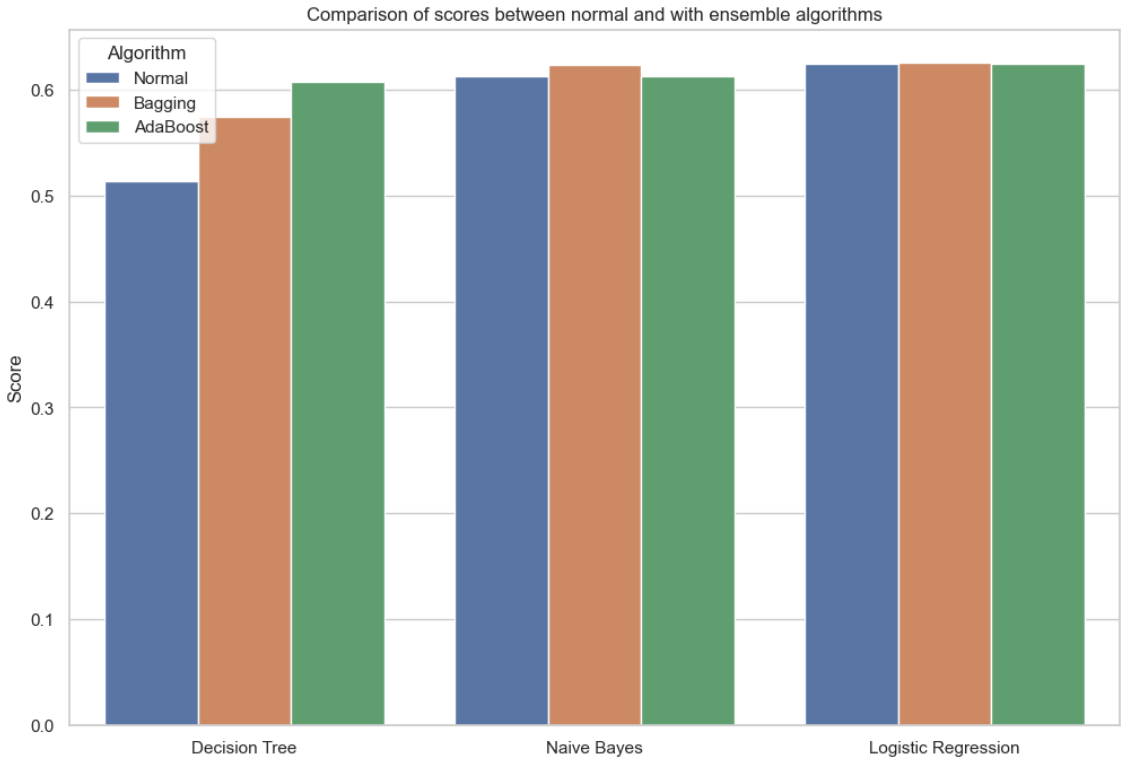

Fig 12. Normal models vs ensemble ones (Multiclass classification)

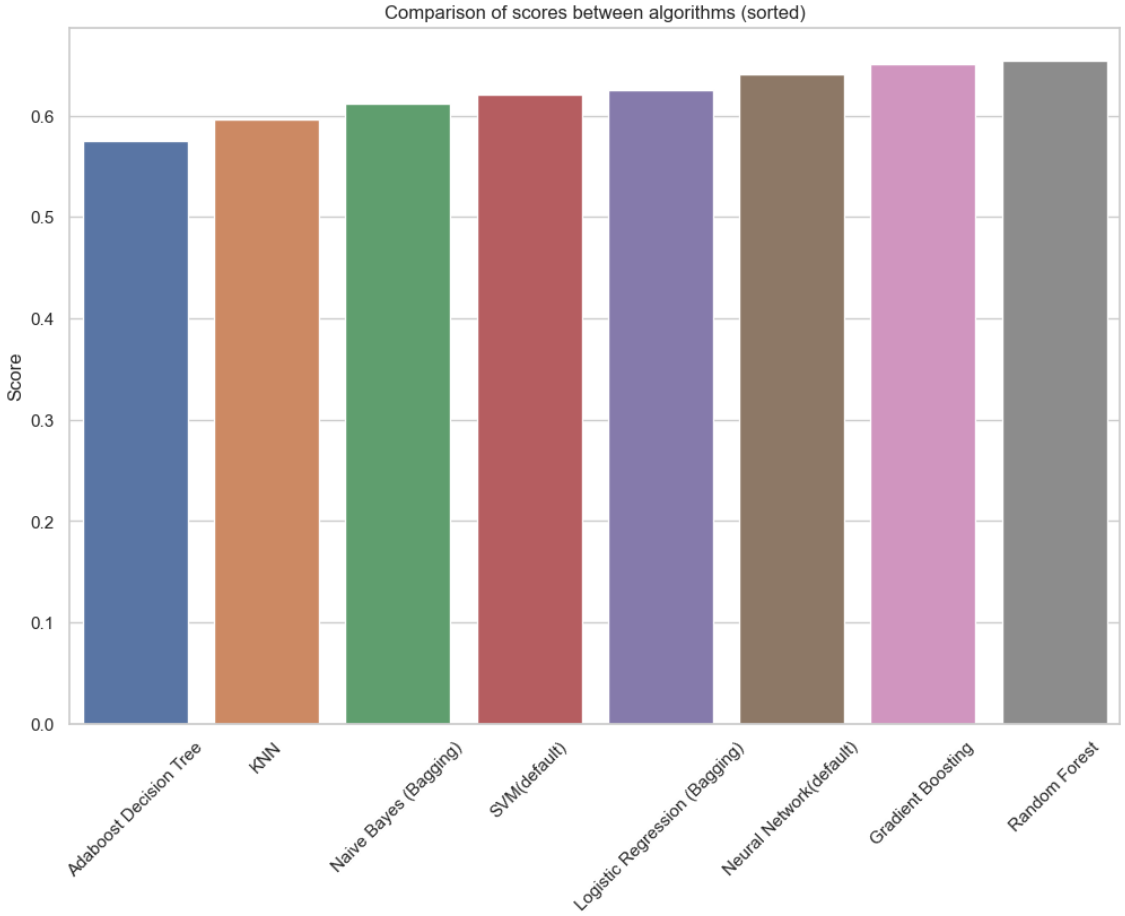

Fig 13. Model performance (Multiclass classification)