Índice

Introduction

Readmissions are considered an important indicator of healthcare-quality and cost-efficiency in every hospital. If a significant number of patients is readmitted in a short period of time after being discharged, there is evidence of potential issues related with the healthcare services. These kinds of issues can have a significant impact on the costs associated with the services provided. In particular diabetic people have a high impact on financial costs.

Considering what was said before, the main objective of this research is answering the question: " How can we predict if a patient will be readmitted?", and for that we will use the predictive power of machine learning techniques that are within our reach. Regarding the main guidelines of machine learning research, our team designed and implemented the following phases to complete the project: (1) Data Exploration and Visualization, (2) Data Pre-processing, (3) Modeling and to finish (4) Conclusions regarding the best scores obtained during the modeling and prediction results.

A particular step that we may highlight in this project is the existence of 2 types of Classification (Binary (readmissions within 30 days of being discharged) and Multiclass ("No", "<30 days", ">30 days") classification for hospital readmissions) which required a different evaluation between them, particularly related with model assessment and final conclusions.

Before exploring the data available for the predictions, by doing some research we find similar projects on the Kaggle platform that also aim to predict hospital readmissions having in consideration similar variables. With this we gained inspiration about what kind of transformations we may do to our data, namely what variables need to be dropped and why, what categories need to be grouped, and what types of combinations with visualizations can be done (Check references: [1]).By checking also the comments about the steps developed in these projects we were able to gain some knowledge about the way we should perform some steps. A specific case is related with the SMOTE/Resampling application, we find out many comments clarifying that the smote should be applied for class imbalance situations and only for the train data set, because on the other hand if we apply it also to the test data set, we will be converging to an over-optimistic solution since we would test our results with a balanced data set. In the case of our project, we were presented with a similar situation and so when performing the resampling techniques that we will describe later, we considered doing it after splitting the data set, and only to the train data set.

Data Exploration and Visualization

Firstly, our team searches to understand the business that we are dealing with, looking for the main objectives and a trail to follow. We started to import the excel given to us into a jupyter notebook file and then we performed the basic steps of data exploration. We verified that the data is composed of **71236 rows and 31 columns**, the columns correspond to the data-set variables, 20 of them are categorical and 11 of them are numeric. By checking the .info() method we get an overview of the data types of our variables that in general seem to be correct, we also reached some conclusions regarding the number of missing values available in each one of the variables (Table 1 - Annexes).

By looking at the first rows of the data we also identified some **incoherence** regarding the existence of **'?'** that should be classified as missing values (namely in weight and payer_code variables). For these incoherence, the values **'[ ]'** present in the variable '**medication'** and also for the values '**Unknown/Invalid**' present in the variable **'gender'** we convert it all into **' Nan'** values. After these transformations we performed the percentage of 'Nan' presented in each variable to understand the impact of them. We noticed that some variables such as **'weight', 'payer_code', 'medical_specialty', 'glucose_test_result', 'a1c_test_result'** have a significant percentage of missing values *(Table 2 - Annexes)*.

Afterwards there was the need to compute some stacked bar charts Count Plots to understand how the data is distributed among the individuals presented in our data set:

- By combining **'age'** and **'readmitted binary'**: We conclude that there are more older people going to the hospital and that these are the ones with the highest number of readmissions (between 70 and 80 years), maybe due to the common pathologies that appear at this age *(Chart 1 - Annexes)*.
- By combining **'race'** and **'readmitted binary'**: It's evident that the most common race is Caucasian and obviously it´s associated with the highest number of readmissions (*Chart 2 - Annexes)*.
- For the **target variables** (Binary and Multiclass): We verified that in both the classes are extremely unbalanced *(Chart 3/4 - Annexes)*.

We also computed the combination of boxplots for all numerical variables, and we concluded that the variables have true possible outcomes and so even if we have extreme values, they make sense for the analysis. Only the **'average_pulse_bpm'** variable appears to have no outliers, the rest of the variables account for extreme values *(Chart 5 - Annexes)*.

Data Split

To split our data into train and validation data sets we applied Stratified K Fold (since this technique wasn't applied during practical classes we provided an additional explanation on the annexes part of the self-study). **Note that we applied this technique in both binary and multiclass classification, since the target variables differ from one type of Classification to another.**

Pre-processing

Pre-processing respects the transformation of raw data into a format easily understood and analyzed by machine learning techniques.

The first thing we did was **setting the 'encounter_id' variable as the index** since it is unique, the count of **'encounter_id'** corresponds exactly to the number of rows available in the data set. On the other hand **'patient-id'** is not unique, it can be repeated among several rows because one patient can have multiple encounters in the hospital. We will leave this variable in our data set to evaluate if it can be useful later.

Duplicates

Using the .duplicated() method we check the existence of duplicates in the data set, and among all columns we identified 0 rows with duplicates.

Missing values

Our group decided to study each variable metadata in order to understand what can be considered for the missing values identified in each variable.

- For **'payer_code'** according to the metadata we assume that all the missing values are associated with the individuals that have no insurance associated with the hospital services and so we substituted all the missing values by 'No Insurance'.
- For **'medical_specialty'** and **'weight'** we substituted the missing values by 'Unknown'.
- For **'admission_source'**, **'admission_type' and 'discharge_disposition'** we assigned the missing values to 'Not mapped' considering the other categories that already exist in these variables.
- For ´**glucose_test_result**´ and ´**a1c_test_result**´ we assigned the missing values to 'Not tested' because if we don't have any information, it should be considered as it wasn't tested (assumption determined by project description).
- For **'medication'** knowing that '[]' were associated to missing values, we considered that in these cases no medication was taken.
- For **'primary_diagnosis', 'secondary_diagnosis' and 'additional_diagnosis'** we assigned all the missing values to 'No Diagnosis', we assumed that no diagnoses were taken.

At this step only 3 variables remain with missing values: **'race', 'gender' and 'age'.**

For these 3 cases we apply **mode imputation,** considering that these variables do not represent a significant percentage of missing values and in this way, using mode, our values to impute won't be

influenced by outliers. Note that for validation and test data sets, we applied the imputation using the variable's modes of the train data set.

We applied these techniques in both binary and multiclass classification.

Outliers

Understanding outliers is crucial as they can adversely impact model performance and compromise the generalization of the model.

As we said before in the exploration phase, we observed certain extreme values on our data set.

Since we will not use techniques in the imputation of missing values that are influenced by outliers, such as mean imputation, we continued the project without treating them. Later on, our group tried to treat the outliers in the training set, however the **F1 scores** were **lower** than if we had not treated them, so our final decision was to **leave outliers in the original format**.

Aggregation of Categories

We applied this step in the sense that some of the variables were composed by a high number of categories which can generate problems regarding the efficiency of some predictive methods applied.

The variables considered to this step were the following ones:

- **'primary_diagnosis', 'secondary_diagnosis' and 'additional_diagnosis' -** Doing some research we understand that to obtain more insights regarding the diagnosis, we should transform all the codes into the description of the corresponding diagnosis. In this way we substitute the codes by the disease category associated, creating 3 new variables associated **'primary_diagnosis_category', 'secondary_diagnosis_category' and 'additional_diagnosis_category'** (check references: [2] to access the link associated with the disease categories).
- **'admission_type',' admission_source'** - We associate all 'Not Available' values into 'Not Mapped' considering the assumption that they mean the same, that the value is unknown.
- **'payer_code'** - For this variable, considering the high percentage of missing values we decided to integrate all the codes available into one single category. Knowing that what matters the most is if the patient has insurance or not, we associated all the codes into a single category called 'Insurance'. Now this variable has only two possible outcomes: 'Insurance' and 'No Insurance'.
- **'age' -** For this variable we decided to merge some categories, associating more meaningful names, note that we take into consideration the balance between categories *(Image 1)*.
- **'Race' -** Knowing that Caucasians are the dominant race of our data set we change the variable categories into **'Caucasian', 'Not Caucasian'** this last one associated with the remaining races of our data set**.**

We defined a threshold so that we are able to adequately and balance the categories available in each variable, in this way we facilitate the interpretation, providing a lower number of balanced categories, assigning every non relevant class to the **'Other'** common category. Note that we evaluate each variable case in particular to set the correct threshold and prevent any loss of information. **Note that we applied these transformations in binary and multiclass classification.**

Feature Engineering

Creating new variables from existing ones allows us, among other things, to capture non-linear relationships and/or more complex relationships and improve the robustness of the model. Thus we have created the following variables: **'pulse_category'**, **'multiple_diagnosis'**, **'diagnosis_category'**, **'medication_complexity'**, **'lab_test_intensity'**, **'total_visits'**, **'lenght_of_stay_category'**, **'emergency_visit'**, and **'encounter_patient'**. The vast majority of these new variables help us categorize values of the original variables in order to better interpret the condition and specific situation of each individual. The description of each new variable can be found in *(Table 3 - Annexes)*. **Note that we applied these transformations in both binary and multiclass classification.**

Target Encoding

After searching for advantages and disadvantages among all the encoding techniques available we decided to proceed with 'Target Encoding' and a further explanation about this decision is provided in the annexes. **Note that we applied this technique in both binary and multiclass classification.**

Scale the data

Scaling numerical data is a crucial preprocessing step in machine learning to ensure features are on a consistent scale. This is vital for algorithms sensitive to feature magnitudes, such as KNN, improving convergence and preventing numerical instabilities. Scaling also enhances regularization effectiveness and model interpretability by making coefficients comparable.

In our context, **we applied RobustScaler to both binary and multiclass classifications**. This technique is mainly used when maintaining outliers in the dataset (something that happens in our case). Its approach involves standardizing the features, removing the median, and scaling the data according to the interquartile range ((X - XMedian) / (Q3 - Q1)). Of course, we considered other factors to support our decision regarding the type of scaling used.

We compared the scores among different scaling methods, **['MinMax[0-1]', 'MinMax[-1,1]', 'Standard', 'Robust']**. We concluded that there was no significant difference between them. Thus, the support for our solution comes mainly from other factors, such as the need to ensure that the data is normally distributed for Standard Scaling, something we cannot guarantee in our case. And on the other hand, using MinMax scaler does not reduce the importance of outliers. Due to the previous reasons, we discard the use of MinMax and Standard scalers.

Feature Selection

This important step is focused on the reduction of data set complexity in order to search for the important variables that most contribute to the predictions. Note that before this step we already dropped the 'country' variable due to the fact that it has no variation; the data is only associated with one country, 'USA', it is a constant feature. We also dropped **'pattient_id'** considering the fact that at this point does not add any relevant information to our model.

Binary Classification

The first technique we applied in feature selection was **Chi-square Test of Independence** to check, before applying the target encoding, which categorical variables are relevant for the target predictions.

Only one variable was considered to be not important, **'weight',** and the high percentage of missing values had probably contributed to this conclusion.

After performing the target encoding, we **look at the variation of each variable,** and we check that every variable has in some way variation, although some variables such as **'payer_code',** which do not have a higher number of different categories, have lower values.

Next to the previous e step we applied **Filter methods for all existing variables of our data set.** In our case we decided to use the spearman method instead of the Pearson method because using 'spearman' we do not need to ensure any assumption (normality, no outliers, random sample).

By analyzing the **Spearman Correlation matrix** we conclude that obviously some new variables created later on by our team are extremely correlated with the initial ones. We considered all the variables with correlation higher than **0.5** for the decision of being discarded *(Image 2 - Annexes)*.

We also applied **Wrapper methods** to gain more insights regarding the decisions that we need to make in this step. The first method that we applied was Recursive Feature Elimination **(RFE) conjugated with the Logistic regression model by evaluating F1 Scores results.** Note that we applied RFE only to our initial numeric features to facilitate the running time. This method only considered the **'encounter_patient'** variable to be important with a weighted F1 score in the validation data set of 84%.

Finally, we also take a look at **Embedded methods,** by applying **Lasso Regression** in order to study the coefficient importance for each variable in our data set. **Lasso picked 12 variables and eliminated the other 23 variables, the ones with coefficients assigned to 0.**

Multiclass Classification

For multiclass classification we also applied the **Chi-square Test of Independence,** however the results were not the same. All the categorical variables were considered to be important at this time. Next to this step, we evaluated the **correlation matrix following the same approaches that we mentioned previously**, and the conclusions seem to be the same we saw before in the binary classification problem. Moving on to the Wrapper methods our group performed again **RFE, however since we are dealing with a multiclass target we had the need to specify some parameters when defining the Logistic Regression model (multi_class='multinomial', solver='lbfgs', max_iter=1000).** The model

results advised us to retain 3 features, considering that the Weighted F1 score with 3 optimal features on validation set was about 0.46, a quite moderated performance. Lastly, we performed Embedded methods, **Lasso picked 21 variables and eliminated the other 14 variables**

By comparing the variables dropped in Binary and Multiclass Classification we conclude that in Multiclass classification the feature selection methods were not so selective about the features to drop, and this can happen due to the increase of complexity in Multiclass Classification problems.

The final decisions of the feature selection process were all combined into 2 single tables *(Table 4 (Binary Classification) and Table 5 (Multiclass Classification) - Annexes)*.

Resampling

After performing feature selection by dropping non relevant features our team was able to proceed with the modeling process. We started by performing a Logistic regression model and to support our analysis we computed the summary metrics between predictions of train and validation data sets. (Check Image 4 - Annexes) We mainly understand having a higher accuracy does not mean good prediction because in this case due to the lower number of 'yes' in our dataset we are not predicting it with a good **precision or recall.**

As a consequence we don't have a good F1 score (about 0,07 in train and 0 in validation) and that's why we need other techniques to balance our data set, such as resampling. As resembling techniques were not covered during classes we provided an additional explanation about the topic in the annexes. Due to the reasons explained in the annexes we proceed with the undersampling solution. **Note that we applied these techniques in both binary and multiclass classification.**

Modeling

After applying the resampling method mentioned previously, we started to perform the models learned during classes and we noticed a significant improvement in the general F1 scores. Our focus from this point on was to improve this score, having in mind that it should be evaluated when seeking for a balance between Precision and Recall and if there is an uneven class distribution (a large number of actual Negatives, which is our situation).

For some methods we have many options of parameterization and to optimize this kind of choices it's a common action to use a grid search to determine among the available parameters which ones are the best ones to use. The best parameters to use are the ones that are less prone to overfit.

In other words, grid search involves defining a grid of hyperparameter values (many options) and searching through all possible combinations of these values to find the set of hyperparameters that results in the best model performance.

Binary Classification

Bayes Classifier

This is a classification model that is based on the Bayes Theorem. It focuses on the calculation of the most probable prediction of unseen observations. We considered the assumptions of independence between features that we checked previously using Chi Square test in Feature Selection and also the process of scaling the data.

With this model we reached an F1 score associated with the 'Yes' label of 0.49379 in train and 0.246 in validation, having evidence of overfit.

Logistic Regression

This model was designed to predict the output of a dependent variable. The

outcome values must be categorical or discrete, such as "Yes" or "No", which happens in our case. We also specify **(random_state=42, max_iter = 400)**.

With this model we reached an F1 score associated with the 'Yes' label of 0.61541 in train and 0.26351 in validation, having evidence of overfit.

K-Nearest Neighbours Classifier

Since this model has a variety of parameters that can be chosen, we decided to apply Grid Search in order to optimize this process. We applied this model using the 'kd_tree' algorithm and the parameters obtained during the Grid Search **(metric='euclidean', n_neighbors=11, weights='uniform')**

With this model we reached an F1 score associated with the 'Yes' label of 0.71639 in train and 0.25283 in validation, having evidence of overfit.

Neural Networks

For this model we follow the same approach said before, using Grid Search to choose the best parameters. Those parameters were the following ones: **(max_iter=400, solver= 'adam', learning_rate= 'adaptive', hidden_layer_sizes= 10, activation= 'relu').**

With this model we reached an F1 score associated with the 'Yes' label of 0.7054 in train and 0.30122 in validation, having evidence of overfit.

Decision Trees

For the decision tree model our group applied what is called preprunning the parameters before building the tree. And in this way, we prevent the tree from becoming too complex and overfitting the training data. For this process we decided to apply Grid search again, finding that these were the best parameters **(criterion= 'entropy', max_depth= 5, splitter= 'best' , min_samples_split= 5, min_samples_leaf = 1 , max_features= None , max_leaf_nodes=10, min_impurity_decrease = 0.0).**

With this model we reached an F1 score associated with the 'Yes' label of 0.70343 in train and 0.28091 in validation, having evidence of overfit.

Random Forests

Moving on to ensemble learning we start by performing Random forests. **By setting bootstrap=True, random_state=42 we ensure diversity and reproducibility respectively.**

With this model we reached an F1 score associated with the 'Yes' label of 0.61541 in train and 0.26351 in validation, having evidence of overfit.

Stacking

For stacking we only defined the following base learners: **[('dt', DecisionTreeClassifier(random_state=42)), ('svm', SVC(random_state=42))]**

With this model we reached an F1 score associated with the 'Yes' label of 0.70801 in train and 0.28037 in validation, having evidence of overfit.

AdaBoost Model

For this model we also used Grid search, and with this technique we considered the following parameters to be the best ones **(random_state=42, n_estimators= 200, learning_rate= 0.1, estimator= DecisionTreeClassifier(max_depth=2)).**

With this model we reached an F1 score associated with the 'Yes' label of 0.71574 in train and 0.29626 in validation, having evidence of overfit.

Gradient Boosting

This model combines multiple weak models to get better performance. In

in other words, intends to minimize the overall prediction error. For this model we also used Grid search, and with this technique we considered the following parameters to be the best ones **(random_state=42, subsample= 0.8, n_estimators= 300, min_samples_split= 4, min_samples_leaf= 2, max_features= None, max_depth= 5, learning_rate= 0.01)**

With this model we reached an F1 score associated with the 'Yes' label of 0.72860 in train and 0.30166 in validation, having evidence of overfit.

Multiclass Classification

For multiclass classification we may highlight the use of weighted F1 score in all model evaluations, since in our case having an imbalanced dataset the weighted average is preferred. In other words this type of average gives more importance to classes with larger support, reflecting the overall performance of the classifier while considering class imbalance. For multiclass classification we also applied grid search in some models, the same ones of Binary Classification. However as the target variable changes the parameters classified as the best ones for Binary Classification will naturally be different for the multiclass problem. To summarize the results we must state that the scores of the models in general were good, we only noticed some evidence of overfitting in **K-Nearest Neighbours Classifier, Stacking Ensemble, Gradient Boosting and Random Forest models** (Table 7 - Annexes)**.**

Conclusions

Binary Classification

Even after applying cross validation, grid search in some models and resampling techniques, there is still evidence of overfitting in several models which was not expected (Table 6 - Annexes). However, we proceed to test the various model solutions on Kaggle. Based on Kaggle submissions performance the model that we used to perform our final predictions was **Gradient Boosting**, with this model our team achieved a score on Kaggle's platform about 32%, in comparison with the other groups available for the competition it seems to be an adequate score.

Multiclass Classification

For the multiclass problem, by comparing among all the model approaches that we performed, our team concluded that the best choice would be to perform the predictions using the **Decision Tree algorithm,** since it exhibits a balanced performance between validation/train data sets scores (no evidence of overfitting) and also the highest scores on the validation data set, about 0,54 (Table 7 - Annexes).

To summarize we must state that the findings were achieved, we reached predictions results regarding the models that were considered the best choices for our objective. There is also evidence that we could improve our work progress since there are many other techniques that we can explore (other ways of resampling, encoding), however we considered that with our work we developed the essential steps for the project.

References

[1] *Prediction on Hospital Readmission*. (n.d.). Kaggle.com. Retrieved December 9, 2023, from https://www.kaggle.com/code/iabhishekofficial/prediction-on-hospital-readmission/notebook

[2] *List of ICD-9 codes*. (2019, December 30). Wikipedia. https://en.wikipedia.org/wiki/List_of_ICD-9_codes

[3] *StandardScaler, MinMaxScaler and RobustScaler techniques - ML*. (2020, July 15). GeeksforGeeks. https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml/

[4] *Machine Learning citation style [Update November 2023]*. (n.d.). Paperpile. Retrieved December 9, 2023, from https://paperpile.com/s/machine-learning-citation-style/

[5] Ellis, C. (2023, January 21). *Oversampling vs undersampling for machine learning*. Crunching the Data. https://crunchingthedata.com/oversampling-vs-undersampling/

[6] *Target Encoding*. (n.d.). Kaggle.com. Retrieved December 9, 2023, from https://www.kaggle.com/code/ryanholbrook/target-encoding#Introduction

[7] *Data Transformation: one-hot vs target encoding*. (n.d.). Www.linkedin.com. Retrieved December 9, 2023, from https://www.linkedin.com/pulse/one-hot-vs-target-encoding-samuel-edeh/

Annexes

Self-study

Content 1 - Stratified K Fold Split

Stratified K Fold was the method chosen to split the data since it ensures that each fold has a similar distribution of classes as the original dataset, leading to a well representation of the data and reduction of bias. In this way we ensure that the model is more likely to generalize well to unseen data.

Using this type of split is particularly important when one class significantly outnumbers the other classes (which in our case happened in binary and multiclass classification targets), as it helps prevent models from being overly biased toward the majority class.

Parameters:

- **n_splits:** n_splits=10 indicates that the dataset will be divided into 10 folds for cross-validation.
- **random_state**: It ensures that the same random splits are generated each time you run the cross-validation. In other words ensures the same results every time we need to 'rerun' the code
- **shuffle:** If shuffle=True, the data will be randomly shuffled before cross-validation. This is done to guarantee that each fold represents a random and unbiased subset of the data set. On the other hand if shuffle=False, the data is split into folds without shuffling what can lead to poor solutions.

Content 2 - Target Encoding

Unlike one-hot encoding, target encoding does not lead to high-dimensional sparse matrices, only retains information about the relationship between the categorical variables and the target variable. Thus we consider this option of encoding to be better for future interpretation purposes

In situations where classes are imbalanced, it can provide a more balanced representation by considering the target variable's distribution, which is essential for making accurate predictions.

There is risk of data leakage if the encoding is applied to the entire data set before splitting into training and validation data sets. Thus we only introduce target encoding after doing Stratified k Fold slitting, fitting the encoder only to train data. On validation and test sets we only use the transform method.

We introduce the target encoding method by defining the encoder fitting and transforming the training data with it. And later on we use the same encoder to transform the Validation and Test data.

Content 3 - Resampling

We applied both techniques, Oversampling and Undersampling. Oversampling consists in the generation of minority class instances ('Yeses' in our case) to achieve a balanced distribution. This technique may not be the best choice for imbalanced datasets since the random duplication of minority classes may lead to the introduction of noise into the dataset, as the duplicated instances might not be accurate which can potentially degrade the performance of the model. Unlike Oversampling, Undersampling is a technique that focuses on reducing the number of instances in the majority class (in our case Non readmitted people) to create a balanced dataset that focuses more on the minority class during the training process. Both techniques have advantages and disadvantages, but we consider Undersampling to be the better one to apply as the final decision. Knowing that in our case the dataset is relatively large, undersampling the majority class might be a more computationally efficient option. Oversampling is a better option when we deal with moderated imbalanced data sets of lower dimensions.

Parameters:

- **'sampling_strategy' -** Defined as 1:1 ratio between minority and majority class, sets the counts of 'Yes' equal to the count of 'No', we experiment different sampling strategies and we proceed with the one that gives us the better results.
- **'random_state'** Only to ensure the same division, or results everytime we run the code.
- **'replacement' = True** The same sample may be selected multiple times and, in this way, it ensures diversity, this can compensate for any loss of information during the undersampling application.

Differences in Multiclass classification:

For the Multiclass Classification problem, we applied a different approach only regarding the **sampling_strategy = desired_class_samples**: This parameter is set based on the dictionary desired_class_samples, indicating the desired number of samples for each class. At the end we defined 5000 samples for each class however continuously tried to adjust these values based on the final performance of the models.

Tables

Table 1 *- Variables, counts and d-types*

Variable	Non-Null Count	Dtype
encounter_id	71236	int64
country	71236	object
patient_id	71236	object
race	67682	object
gender	71236	object
age	67679	object
weight	71236	object
payer_code	71236	object
outpatient_visits_in_previous_year	71236	int64
emergency_visits_in_previous_year	71236	int64
inpatient_visits_in_previous_year	71236	int64
admission_type	67530	object
medical_specialty	71236	object
average_pulse_bpm	71236	object
discharge_disposition	68646	object
admission_source	665518	object
length_of_stay_in_hospital	71236	int64
number_lab_tests	71236	int64
non_lab_procedures	71236	int64
number_of_medications	71236	int64
primary_diagnosis	71236	object
secondary_diagnosis	71236	object
additional_diagnosis	71236	object
number_diagnosis	71236	object
glucose_test_result	3688	object

a1c_test_result	11916	object
change_in_meds_during_hospitalization	71236	object
prescribed_diabetes_meds	71236	object
medication	71236	object
readmitted_binary	71236	object
readmitted_multiclass	71236	object

Table 2 *- Missing values percentage*

Variable	Percentage of Missing Values
encounter_id	0,00%
country	0,00%
patient_id	0,00%
race	7,12%
gender	0,00%
age	4,99%
weight	**96,85%**
payer_code	**39,59%**
outpatient_visits_in_previous_year	0,00%
emergency_visits_in_previous_year	0,00%
inpatient_visits_in_previous_year	0,00%
admission_type	5,20%
medical_specialty	**49,02%**
average_pulse_bpm	0,00%
discharge_disposition	3,64%
admission_source	6,62%

length_of_stay_in_hospital	0,00%
number_lab_tests	0,00%
non_lab_procedures	0,00%
number_of_medications	0,00%
primary_diagnosis	0,02%
secondary_diagnosis	0,37%
additional_diagnosis	1,42%
number_diagnosis	0,00%
glucose_test_result	**94,82%**
a1c_test_result	**83,27%**
change_in_meds_during_hospitalization	0,00%
prescribed_diabetes_meds	0,00%
medication	22,95%
readmitted_binary	0,00%
readmitted_multiclass	0,00%

Table 3 *- Feature Engineering*

Variable	Description
pulse_category	Categorizes average pulse rate into 'Low', 'Normal', 'High'.
multiple_diagnosis	Indicates whether there are multiple diagnoses ('Yes' or 'No').
diagnosis_category	Categorizes the number of diagnoses into 'Low Number', 'Medium Number', 'High Number'.
medication_complexity	Categorizes the number of medication into 'Low', 'Medium', 'High', 'Very high'.
lab_test_intensity	Categorizes the number of lab tests into 'Low', 'Medium', 'High'.
total_visits	Toal number of visits.

length_of_stay_category	Categorizes the time spent in the hospital into 'Short Stay', 'Medium Stay', 'Long Stay'.
emergency_visit	Binary varibale that categorizes whether there are emergency visits ('Yes' or 'No').
encounter_patient	encounter_patient' by counting the visits that the same patient made to the hospital.

Table 4 *- Feature Selection Table (Binary Classification)*

Variables / Methods	Chi Square	Spearman	RFE LR	Lasso	Final Decision
weight		x	Not used		Exclude
glucose_test_result	x	x	Not used		Include
a1c_test_result	x	x	Not used		Include
admission_type	x	x	Not used		Include
medical_speciality	x	x	Not used		Include
race	x	x	Not used		Include
gender	x	x	Not used		Include
age	x	x	Not used		Include
payer_code	x	x	Not used		Include
discharge_disposition	x	x	Not used	x	Include
admission_source	x		Not used		Exclude
change_in_meds_during_hospitalization	x	x	Not used		Include
prescribed_diabetes_meds	x	x	Not used		Include
medication	x		Not used		Exclude
primary_diagnosis_category	x	x	Not used		Include
secondary_diagnosis_category	x	x	Not used	x	Include

additional_diagnosis_category	x	x	Not used	x	Include
pulse_category	x		Not used		Exclude
diagnosis_category	x		Not used		Exclude
medication_complexity	x	x	Not used		Include
lab_test_intensity	x		Not used		Exclude
multiple_diagnoses	x	x	Not used		Include
lenght_of_stay_category	x		Not used		Exclude
emergency_visit	x	x	Not used	x	Include
outpatitent_visits_in_previous_year	Not used	x		x	Include
emergency_visits_in_previous_year	Not used	x		x	Include
inpatient_visits_in_previous_year	Not used	x		x	Include
average_pulse_bpm	Not used	x			Exclude
lenght_of_stay_in_hospital	Not used	x		x	Include
number_lab_tests	Not used	x		x	Include
non_lab_procedures	Not used	x		x	Include
number_of_medications	Not used			x	Exclude
number_diagnosis	Not used	x		x	Include
total_visits	Not used				Exclude
encounter_patient	Not used	x	x	x	Include

Table 5 *- Feature Selection Table (Multiclass Classification)*

Variables / Methods	Chi Square	Spearman	RFE LR	Lasso	Final Decision
weight	x	x	Not used	x	Include
glucose_test_result	x	x	Not used		Include
a1c_test_result	x	x	Not used		Include
admission_type	x	x	Not used		Include
medical_speciality	x	x	Not used	x	Include
race	x	x	Not used		Include
gender	x	x	Not used		Include
age	x	x	Not used	x	Include
payer_code	x	x	Not used		Include
discharge_disposition	x	x	Not used	x	Include
admission_source	x		Not used	x	Include
change_in_meds_during_hospitalization	x	x	Not used		Include
prescribed_diabetes_meds	x	x	Not used	x	Include
medication	x		Not used	x	Include
primary_diagnosis_category	x	x	Not used	x	Include
secondary_diagnosis_category	x	x	Not used	x	Include
additional_diagnosis_category	x	x	Not used	x	Include
pulse_category	x		Not used		Exclude
diagnosis_category	x		Not used		Exclude
medication_complexity	x	x	Not used	x	Include
lab_test_intensity	x		Not used		Exclude
multiple_diagnoses	x	x	Not used		Include

lenght_of_stay_category	x		Not used		Exclude
emergency_visit	x	x	Not used	x	Include
outpatitent_visits_in_previous_year	Not used	x		x	Include
emergency_visits_in_previous_year	Not used	x	x	x	Include
inpatient_visits_in_previous_year	Not used	x		x	Include
average_pulse_bpm	Not used	x		x	Include
lenght_of_stay_in_hospital	Not used	x		x	Include
number_lab_tests	Not used	x		x	Include
non_lab_procedures	Not used	x		x	Include
number_of_medications	Not used			x	Exclude
number_diagnosis	Not used	x	x	x	Include
total_visits	Not used				Exclude
encounter_patient	Not used	x	x	x	Include

Table 6 *- Final F1 Scores (Binary Classification)*

Model	F1 Score (Train - Label: 'Yes')	F1 Score (Validation - Label: 'Yes')
Bayes Classifier	0.49379	0.24600
Logistic Regression	0.61541	0.26351
K-Nearest Neighbours Classifier	0.71639	0.25283
Neural Networks	0.70541	0.30122
Decision Trees	0.70343	0.28091
Random Forest	0.61541	0.26351
Stacking Ensemble	0.70801	0.28037
Ada Boosting Ensemble	0.71574	0.29626
Gradient Boosting Ensemble	0.72860	0.30166

Table 7 *- Final F1 Scores (Multiclass Classification)*

Model	Average F1 Score (Train)	Average F1 Score (Validation)
Bayes Classifier	0.4191	0.5120
Logistic Regression	0.5031	0.4654
K-Nearest Neighbours Classifier	1.0	0.4869
Neural Networks	0.5252	0.4990
Decision Trees	0.4990	0.5383
Random Forest	1.0	0.5371
Stacking Ensemble	0.8590	0.4878
Ada Boosting Ensemble	0.5523	0.5264
Gradient Boosting Ensemble	0.8604	0.5268

Charts

Chart 1 *- Stacked bar chart ('age','readmitted_binary')*

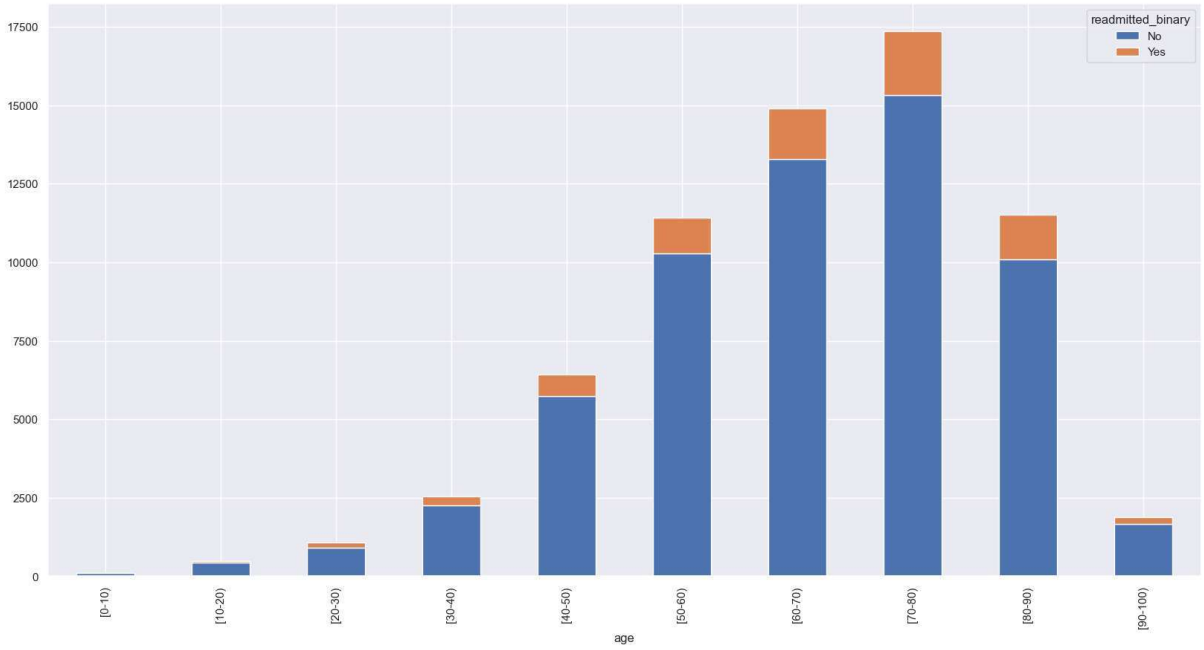


Chart 2 *- Stacked bar chart ('race', 'readmitted_binary')*

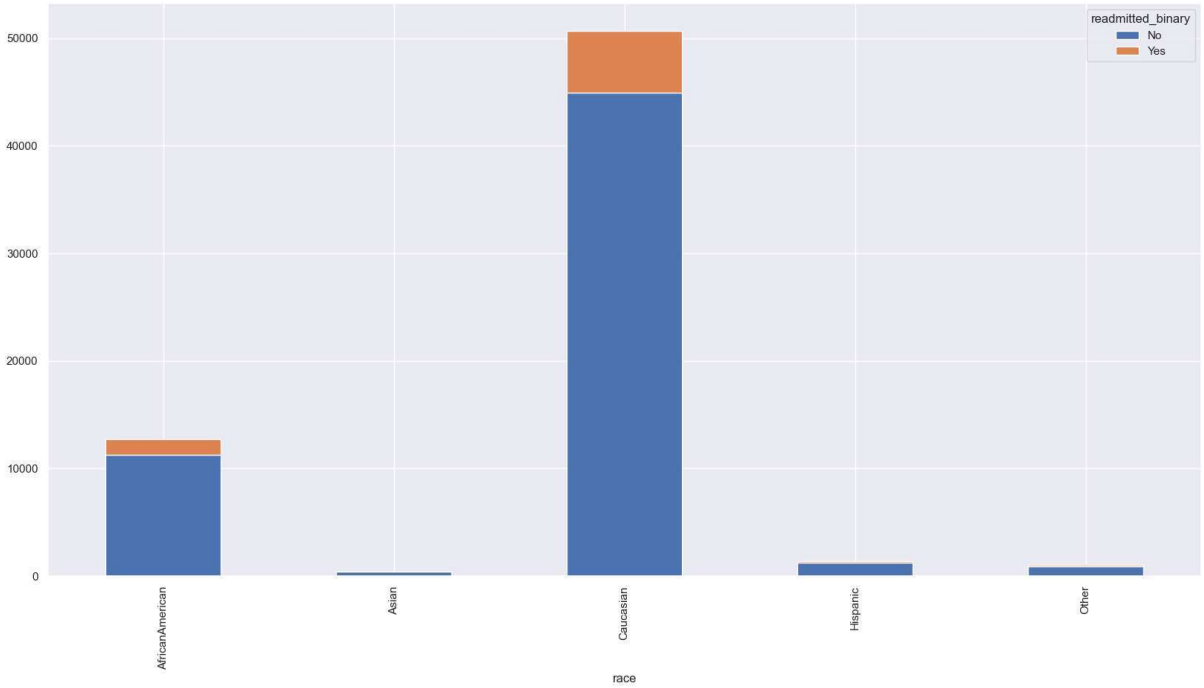


Chart 3 *- Countplot of 'readmitted_binary'*

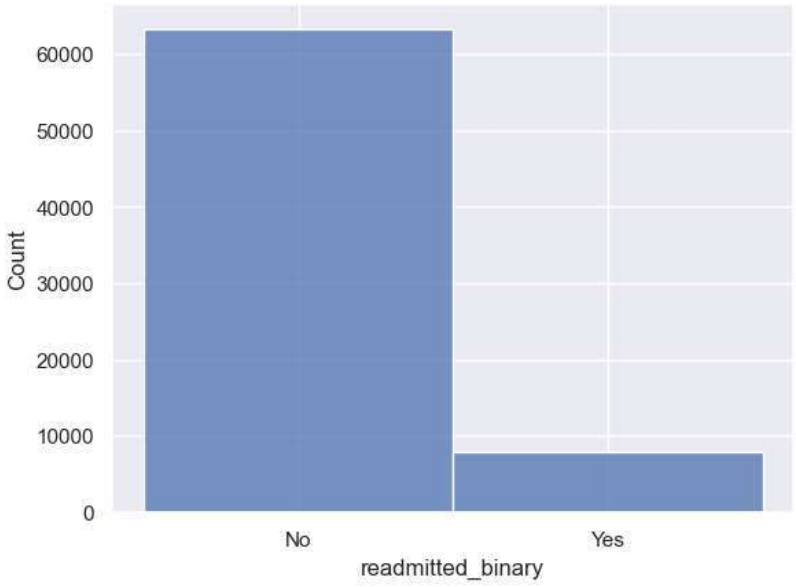


Chart 4 *- Countplot of 'readmitted_multiclass'*

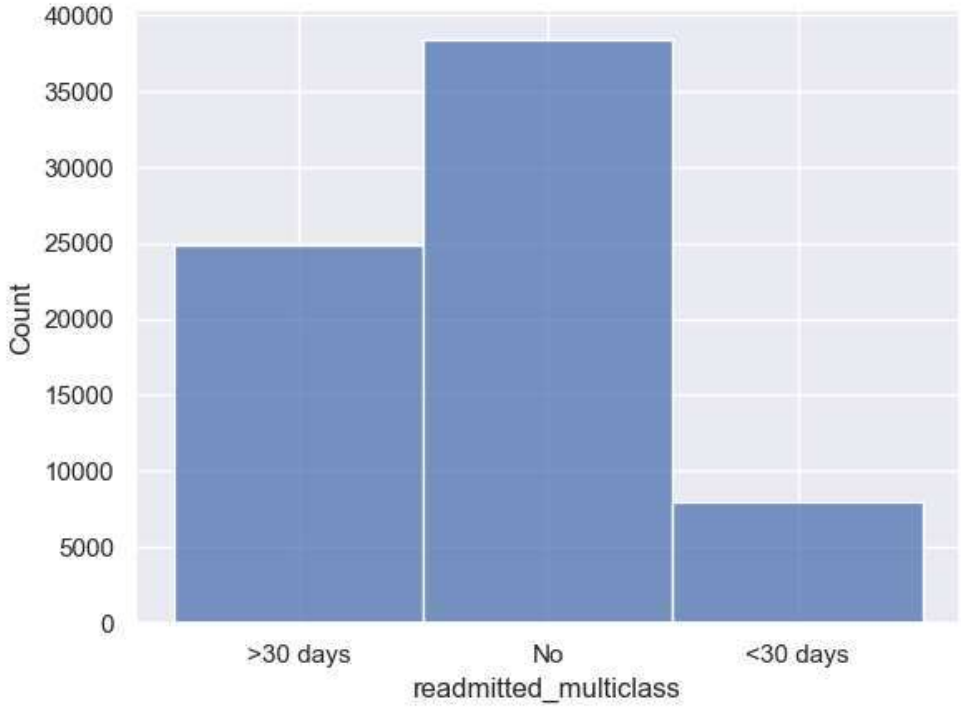
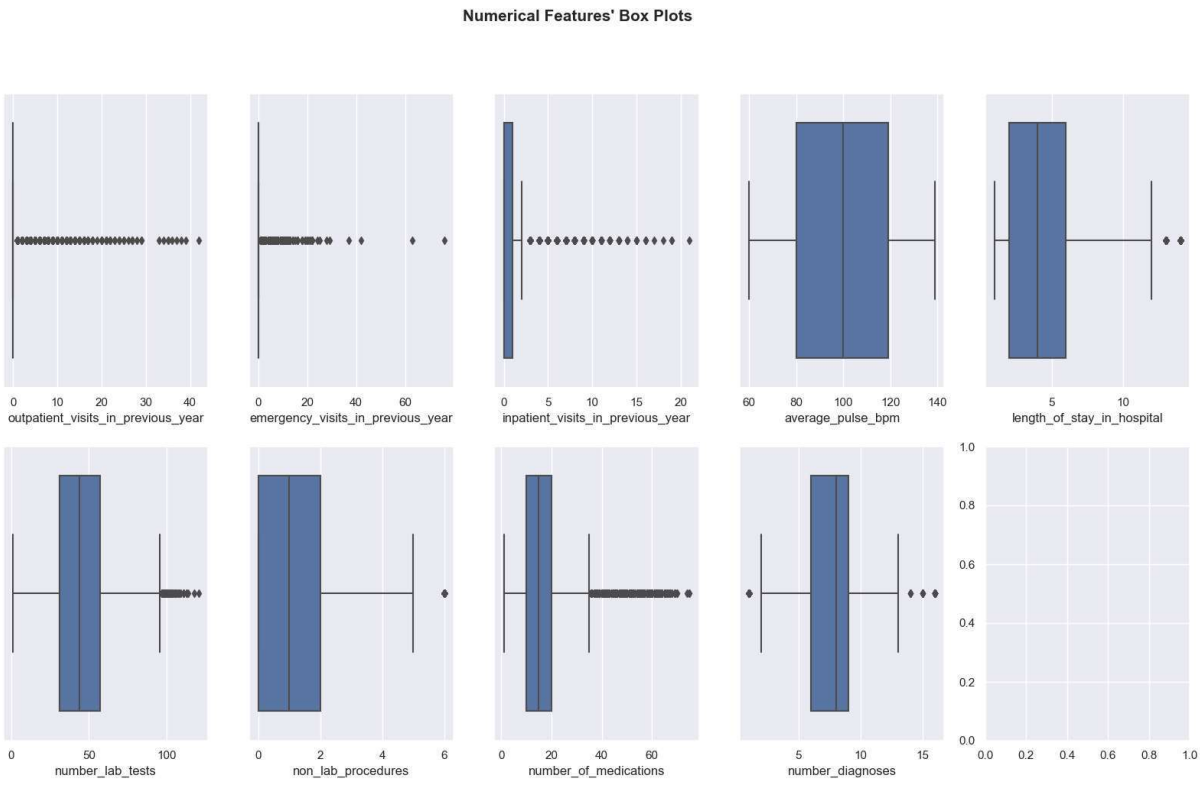


Chart 5 *- Numeric variables Boxplots*



Images

Image 1 *- Age groups - code*

```
# age center
category_mapping = {
    '[0-10)': 'child',
    '[10-20)': 'child',
    '[20-30)': 'Adult',
    '[30-40)': 'Adult',
    '[40-50)': 'Adult',
    '[50-60)': 'Middle Age',
    '[60-70)': 'Middle Age',
    '[70-80)': 'Old',
    '[80-90)': 'Old',
    '[90-100)': 'Old'

}

X_train['age'] = X_train['age'].replace(category_mapping)
X_val['age'] = X_val['age'].replace(category_mapping)
```

Image 2 *- Correlation Matrix (Binary Classification)*

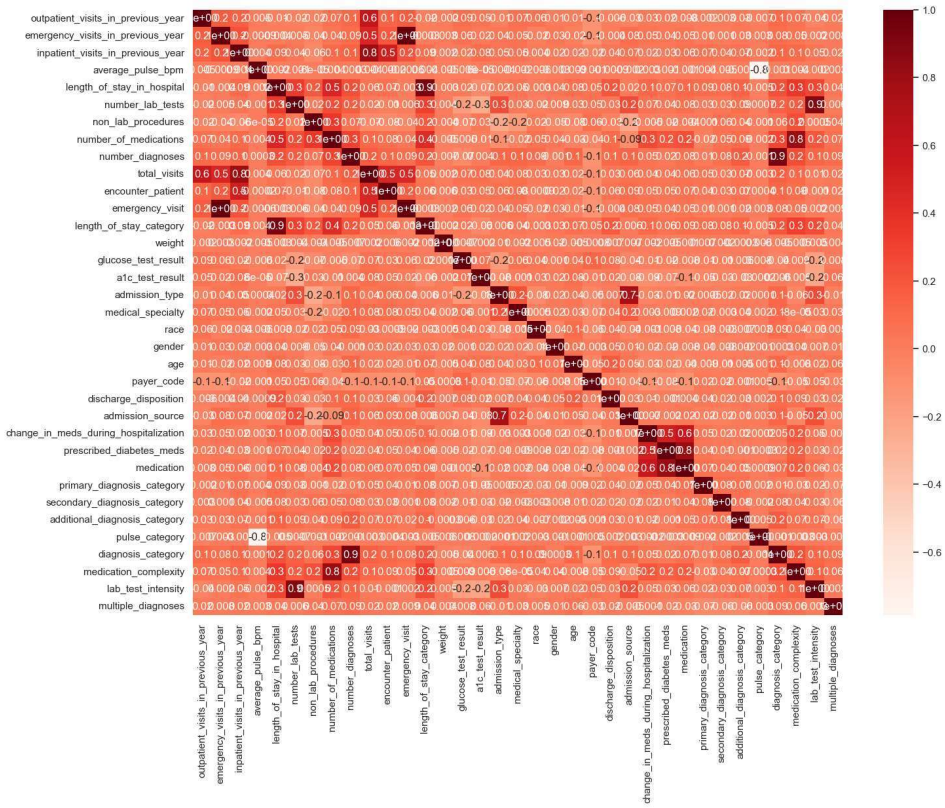


Image 3 *- Correlation Matrix (Multiclass Classification)*

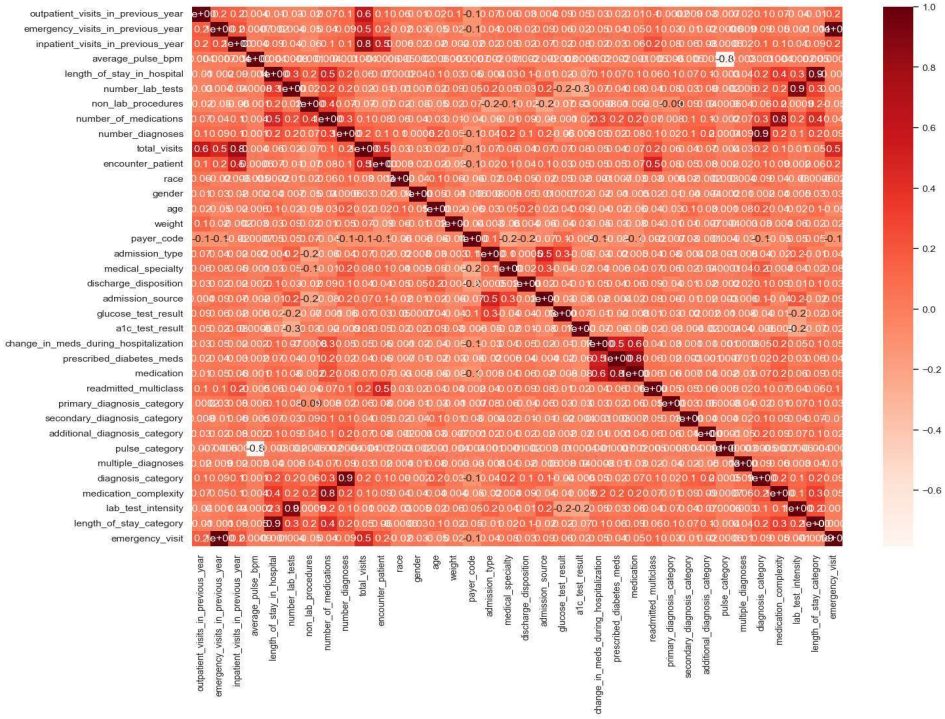


Image 4 *- Logistic Regression without Resampling - Summary Metrics*

---------- TRAIN ----------

	precision	recall	f1-score	support
No	0.89	0.99	0.94	56958
Yes	0.44	0.04	0.07	7155
accuracy			0.89	64113
macro avg	0.67	0.52	0.50	64113
weighted avg	0.84	0.89	0.84	64113

---------- VALIDATION ----------

	precision	recall	f1-score	support
No	0.89	1.00	0.94	6328
Yes	1.00	0.00	0.00	795
accuracy			0.89	7123
macro avg	0.94	0.50	0.47	7123
weighted avg	0.90	0.89	0.84	7123