Predicting Hospital Readmittance with Logistic Regression

Duy Nguyen

1. Introduction

Hospital readmission occurs when a patient has been discharged from a hospital but has been admitted within a specific period. To help promote efficiencies within Medicare, the 2010 Patient Protection and Affordable Care Act established the Hospital Readmissions Reduction Program (HRRP) as an addition to the 1965 Social Security Act. Tracking readmittance can help identify possible ineffective treatments during past hospitalizations and possibly penalize hospitals. It is in the hospital's best interest to identify patients with higher risks of readmission to avoid denied reimbursements and penalties. By identifying high risk patients, additional measures and support could also help lower healthcare costs, improve quality of care, and increase patient satisfaction.

2. Dataset

The dataset, obtained from the Center for Machine Learning and Intelligent Systems at University of California, Irvine, represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes 95,854 entries and 50 features, representing patient and hospital attributes such as patient identification, diagnosis codes, admission type, source, and discharge disposition, risk related medications and test results, and numerous other hospitalization indicators.

2.1. Data Wrangling

Our initial focus consisted of converting '?' values into NULL, removing some under-represented features like 'weight' with 97% of its rows missing and identification columns that don't contribute to modeling results, and imputing the missing values of 'race' proportionally. By observing the proportions of 'race' in Figure 1, there should be a concerning amount of minority in our data in not just 'race', but also in others like 'age'. We equally divided the said missing values of 'race' accordingly to these proportions. These steps have reasonably removed all missing values. Missing values aside, we will tackle data minorities with SMOTE which will be elaborated further later.

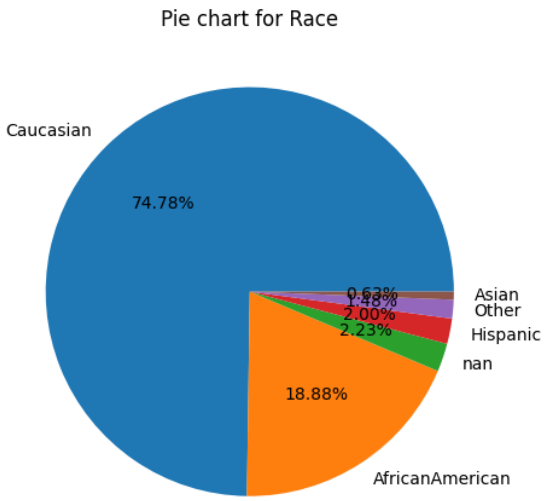

Figure 1. Pie chart of the distribution of race of patients, with Caucasians being the majority and 2.23% being nulls.

Next, the 3 diagnosis codes were correspondingly categorized into their clinical names [1]. And the ' admission_type_id', 'admission_source_id', and 'discharge_disposition_id' all have their codes related to non-admission removed. Finally, the response variable 'readmitted' has been one-hot coded to have, within (less than) 30 days to 1, and others to 0.

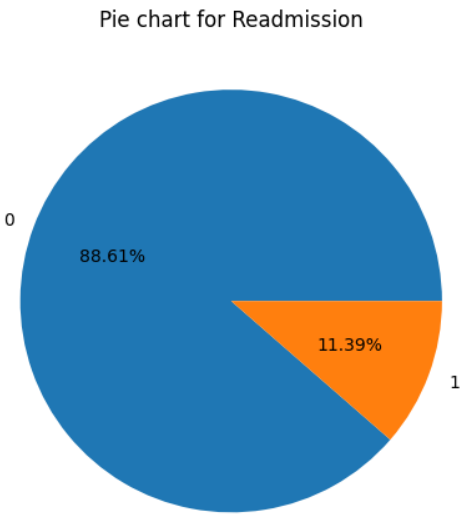

Figure 2. Pie chart of patients who were readmitted within 30 days in orange, versus otherwise in blue.

Not all numeric features should be inputted in the models as numeric, therefore we gathered the numeric features that should be categorical like the 3 diagnosis codes, 'admission_type_id',

'admission_source_id', 'discharge_disposition_id', the variables that are to be one-hot coded like medicine types, and the naturally categorical variables. And likewise, the rest of the numeric features are grouped normally.

Clarified Categorical and Numeric Features

```
categorical_features = ['race', 'gender', 'max_glu_serum', 'A1Cresult',
                        'metformin', 'repaglinide', 'nateglinide',
                        'chlorpropamide', 'glimepiride', 'acetohexamide',
                        'glipizide', 'glyburide', 'tolbutamide',
                        'pioglitazone', 'rosiglitazone', 'acarbose',
                        'miglitol', 'troglitazone', 'tolazamide',
                        'insulin', 'glyburide-metformin',
                        'glipizide-metformin', 'age',
                        'glimepiride-pioglitazone',
                        'metformin-rosiglitazone',
                        'metformin-pioglitazone', 'change', 'diabetesMed',
                        'diag_1', 'diag_2', 'diag_3',
                        'admission_type_id', 'discharge_disposition_id',
                        'admission_source_id']

numeric_features = ['time_in_hospital', 'num_lab_procedures',
                    'num_procedures', 'num_medications',
                    'number_outpatient', 'number_emergency',
                    'number_inpatient', 'number_diagnoses']
```

Figure 3. The clarified lists of both categorical features and numeric features for inputting into models.

2.2. Exploratory Data Analysis

Plots were created for the features that we thought would most influence a patient to be readmitted, to set deeper roots for our predictions in this study.

As seen below, ID = 1, Discharged to home, is most dominant in the data, followed by IDs 3 and 6 equally, which are transferred to Skilled Nursing Facilities and to home with home health services, respectively. Out of the three mentioned, being discharged to home with home health services might be the sole influencer, but there might be other stronger ones from other features, or that the minority classes in this feature are under sampled.

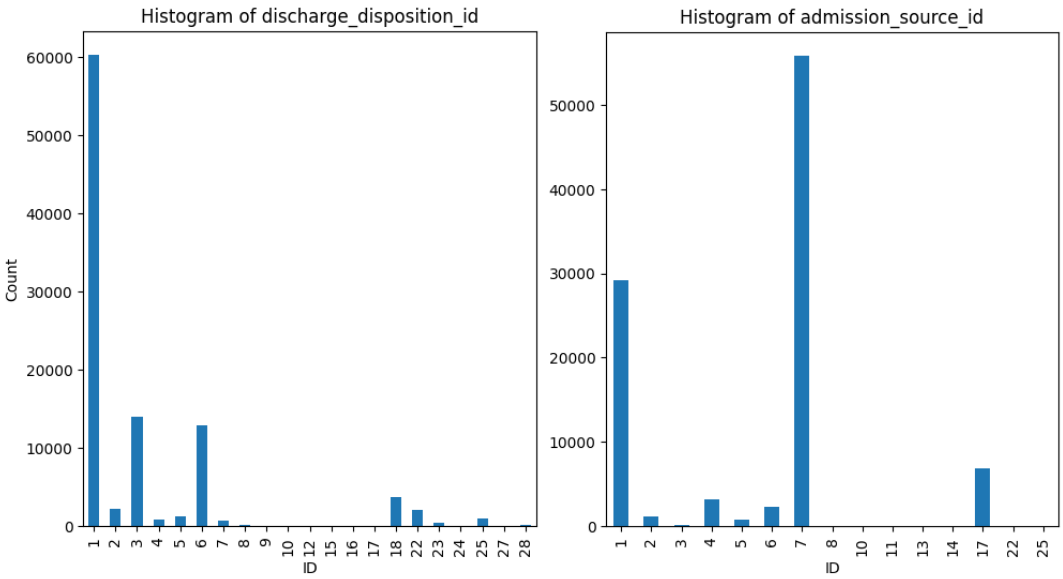

Figure 4. The under-sampled distributions of “discharge_disposition_id” and “admission_source_id”. Most of the IDs are minority classes.

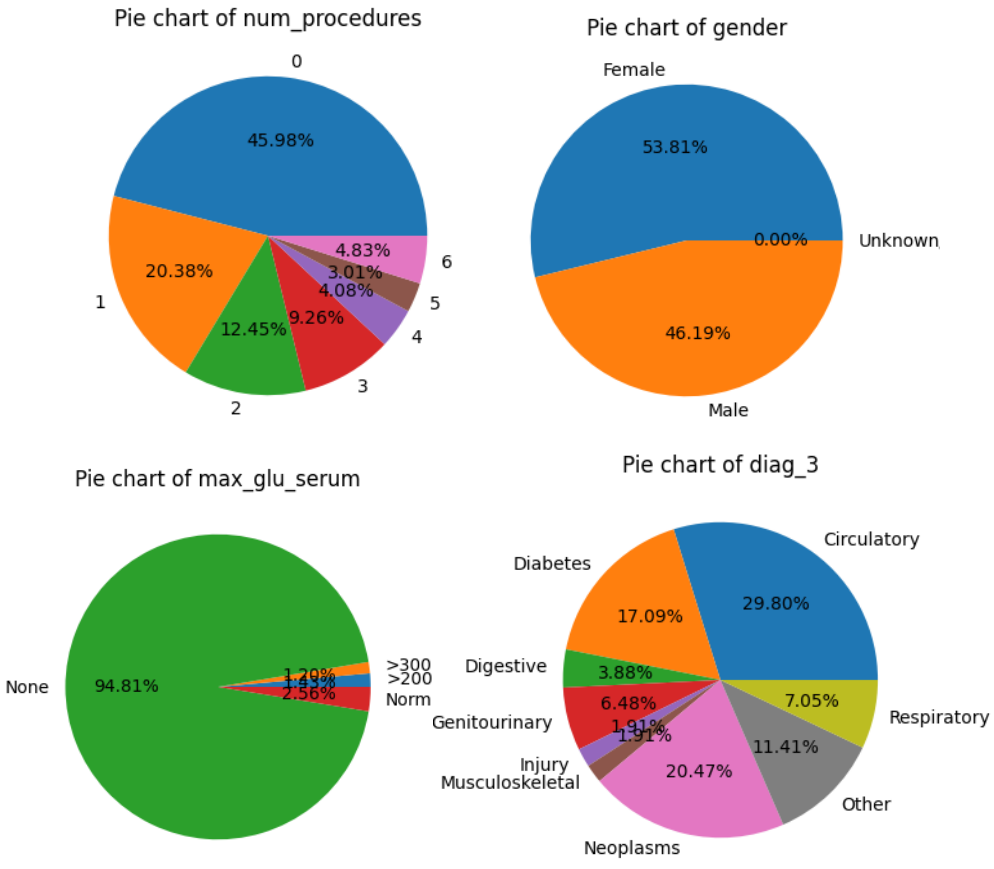

Figure 5. The pie charts of important features chosen with instinct.

3. Logistic Regression Models

The grid search method was performed to find the best hyperparameter to select the most important variables with LASSO (L2) or to suppress the less important variables with Ridge (L1) regularization method.

3.1. Model Building

A general preprocessor pipeline was made to prepare the logistic regression models with random_state = 11. A train-test split was done with 70/30, and the numeric features are processed with RobustScaler(), categorical features are processed with OneHotEncoder(), and stratified k-fold with K = 10. Both L1 and L2 were performed with the general 7 regularization strengths from 10^{-4} to 10^{-2}.

For the first competing model, we used SMOTE to transform the data. Since L1 seems to fit better for this logistic model at alpha = 0.01, the L1 is then tuned further with equal strengths from 0.001 to 0.1, which leads to the 3rd row of Table 1. For the second competing model, we did not use SMOTE to explore, later in this paper, how it impacts which threshold to use to optimize recall and precision. Since L1 seems to fit better for this logistic model at alpha = 0.1, the L1 is then tuned further with equal strengths from 0.01 to 1.0, which came out unsatisfactory, then a second time with equal strengths from 0.09 to 0.2, which leads to the 5th row of Table 1.

3.2. Competing Models

We used the ROC-AUC scoring to score our models and validated the predictions with the test set that was split earlier. The higher the AUC, the better the performance of the model at classification. We did not use accuracy to evaluate the model because the data is very imbalanced, which lead us to using SMOTE, or Synthetic Minority Over-sampling Technique. We use it to oversample the minority class, which happens very often in our diabetic data. SMOTE is a type of data augmentation that synthesizes new samples from the existing ones.

Competing Models

	CV-Score	Test Score	Alpha
SMOTE L1	0.6461	0.6538	0.01
SMOTE L2	0.6459	0.6524	0.001
SMOTE Tuned L1	0.6493	0.6556	0.003
Non-SMOTE L1	0.6598	0.6684	0.1
Non-SMOTE L2	0.6586	0.6681	0.1

Non-SMOTE Tuned L1	0.6599	0.6687	0.13

Table 1. Table of the competing models between L1 and L2 regularization methods, and SMOTE vs non-SMOTE.

3.3. Applying Domain Knowledge

For medical data, it is very likely that we fail to efficiently predict the minority class due to the heavily imbalanced dataset, therefore we would need to decrease the false negative predictions at the expense of high false positive predictions. To further elaborate on this, a high number of false positives is a leisure for us because, for example, a situation where patients predicted to be readmitted within 30 days but didn't really need to do so should not cause huge concerns. In a health insurance scenario, targeting patients this way can be seen as profitable to the insurance company, since this might be how insurance premiums are increased based on risk of readmission, as well as reassuring for the insured patient, but it is more likely the prior. Likewise, we need to reach the lowest number of false negatives as possible, because a patient predicted to not be readmitted within 30 days but really did need to can lead to huge problems. Reassurance for the patient feels much stronger on this side of the lawn because all opportunities to send patients to the hospital are given, as well as faulty predictions are strongly minimized.

3.4. Evaluation Metrics

This translates to a very high recall and a strong compromise on precision. One way to improve recall while maintaining precision is to use an oversampling technique like SMOTE. As seen in Figure 4, the increased threshold indicates that the classed are more balanced with SMOTE, with recall staying high and precision staying low much longer as the threshold increases compared to non-SMOTE. This allowed us to have more room to adjust the threshold with SMOTE compared to without. The sweet spots for both plots were around threshold = 0.17 and threshold = 0.6, but we decided to go with the latter.

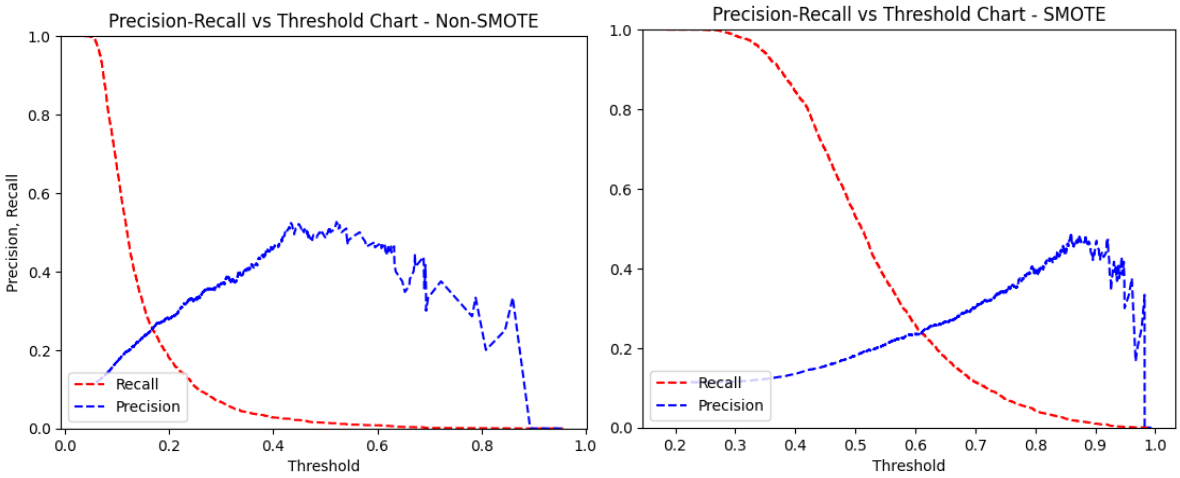

Figure 6. The Precision-Recall plots to show the improvement of class balance between SMOTE versus non-SMOTE. Both precision and recall stay high much longer as threshold increases.

Like we said earlier, we want the lowest false negatives as possible while it's ok for us to have high false positives, while also keeping the F1-score high.

At threshold = 0.48, the final precision-recall values were chosen with classification_report() using predictions on the 30% test set split earlier. Based on the table seen below, we can say that out of all the patients our L1 SMOTE model predicted would be readmitted, only 17% really did. And likewise, out of all the patients that really did get readmitted, our model only predicted correctly for 60% of those patients. Since our F1-Score is quite far from 1, our model isn't quite efficient at predicting readmission.

Classification Report

	Precision	**Recall**	**F1-Score**	**Support**
0	0.92	0.62	0.74	17606
1	0.17	0.60	0.27	2263
Accuracy			0.62	19869
Macro AVG	0.55	0.61	0.50	19869
Weighted AVG	0.84	0.62	0.69	19869

Table 2. The results of our classification model that focuses on precision, recall and F1-score @ threshold = 0.48.

As seen below, we are looking at 40% false negatives and 38% false positives. With threshold = 0.48, we are predicting true positives 60% of the time as well as true negatives 62% of the time.

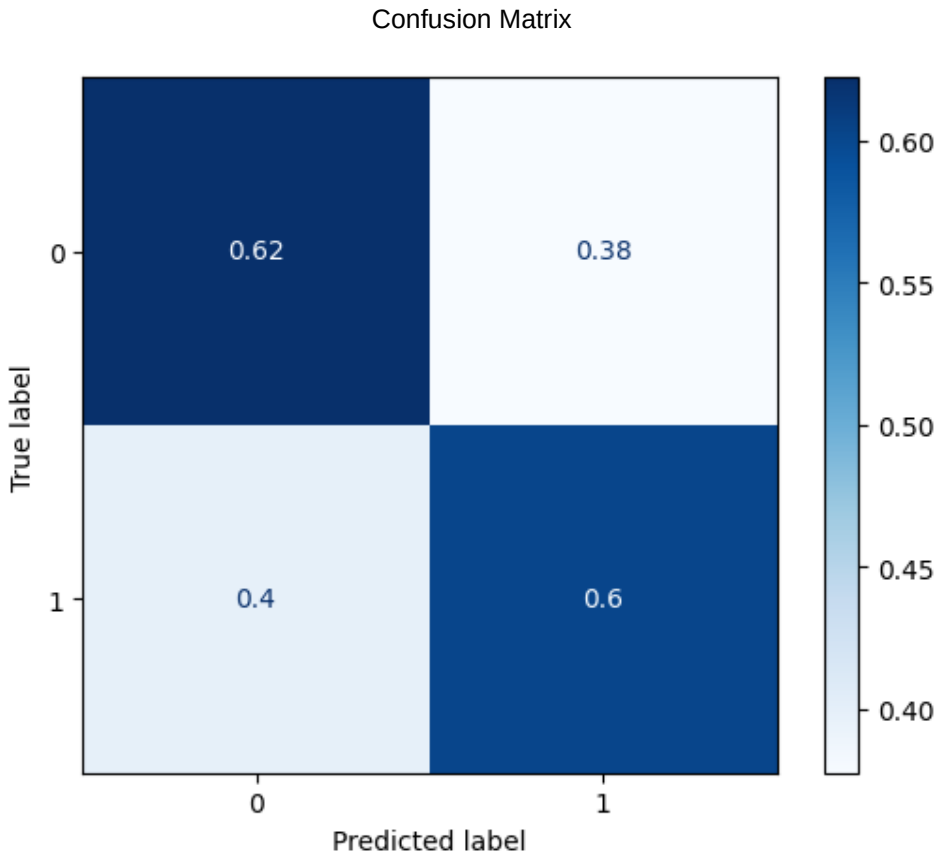

Figure 7. The Confusion Matrix @ threshold = 0.48.

3.5. Feature Importance

The table below shows the top 10 features that contribute to the increase of classification in readmission, with 'dischange_disposition_id_8' or discharged/transferred home under the care of a home IV provider, being the most significant contributor to readmission. After feedback from hospital providers, this code was deemed redundant to disharge_disposition_id_1. Disharge_disposition_id_8 has been discontinued since October 1, 2005, in favor of disharge_disposition_id_1. We state this in case additional work is done on newer data sets where model adjustments may need to be made for the change in code usage.

In Figure 4, we can see the discharge_disposition_id distribution is skewed to the right. ID = 1 represented almost 60% of the population while our #1 feature, ID = 8, represented just .1% of the population. With ID = 8 representing patients sent home with further IV treatment needed, it

may suggest the patient was not completely treated and eventually required another admittance to the hospital.

Our second most important feature is admission_source_id_7 represents patients that were admitted after receiving services in the emergency department. It also represents patients that received unscheduled services in the emergency department and discharged without an admittance. Patients requiring emergency treatment can suggest the severity or the patients lack of willingness to seek preventative treatment. Having either a severe case or not willing to seek timely treatment could indicate a likelihood of readmittance within 30 days. Similarly, our fifth most important feature is admission_source_id_8 that represents patients admitted to upon the direction of a court of law, or upon the request of a law enforcement agency representative. Again, indicating the patient was forced and may not have been actively seeking treatment or appropriately taking care of their diabetes.

Our model indicates that female patients are key indicators to their likelihood of being readmitted to a hospital. Female patients are generally more aware of the health of their body and may be more willing to seek aid quicker if they're feeling abnormal.

Our fourth important feature indicates patients not taking metformin-rosiglitazone also seem to have an increased risk of readmittance. We do not have enough background on the drug and its effects to hypothesize how it can increase a patient's likelihood of readmittance, but our model indicates it may need further research.

The number of procedures a patient has undergone seems to also be a high indicator of whether a patient may be readmitted to the hospital. The higher number of procedures does indicate severity of not only their diabetes but other ailments as well. It also could indicate the amount stress their bodies have endured during treatments, in turn making the patient someone who would be more likely to need additional hospital stays.

We have presented our top 10 features but have only discussed the features weighted higher than 0.1. Although they are weighed lower, they are still worth presenting in case further research is needed for insight into additional indicators of possible readmittance.

Top 10 Important Features

	Feature	Weight	Note
1	discharge_disposition_id_8	0.612653	Transferred to home under care of Home IV provider
2	admission_source_id_7	0.269167	Admission by Emergency Room
3	gender_Female	0.268416	Distribution of females are slightly less than males
4	metformin-rosiglitazone_No	0.175598	Medicine combination used to treat type 2 diabetes
5	admission_source_id_8	0.169067	Admission by Court/Law Enforcement

6	num_procedures	0.113014	Number of procedures done
7	max_glu_serum_>300	0.070908	Simple and direct single test for diabetes
8	diag_3_Diabetes	0.060908	Diabetes as one of patient's diagnoses
9	miglitol_No	0.058262	Oral anti-diabetic drug that helps patient breaks down complex carbohydrates into glucose
10	glipizide-metformin_Steady	0.043848	Medicine combination used to treat high blood sugar levels caused by type 2 diabetes

Table 3. Table of the top 10 most important features with corresponding weights and side notes.

4. Conclusion

With SMOTE, the minority class of our response variable, patients who need to be readmitted within 30 days, is naturally overrepresented. Despite getting a very satisfactory high recall and low precision, we as data scientists should be cautious when using the SMOTE method. When using a resampling method, we are showing the wrong proportions of the two classes to the classifier during the training. Several factors such as new breakthroughs in medical technology and changes in people's diet can lead to changes in the unseen data coming from the future, especially in the proportions of the two prediction classes. The classifier can be learned on a regular basis to accommodate these changes so that it does not lose the information of the true class proportions when resampling.

5. References

1. *https://www.hindawi.com/journals/bmri/2014/781670/tab2/*

6. Appendix

```
# random_state = 11

def preproc(df, categorical_features, numeric_features):

    X = df[categorical_features + numeric_features]

    Y = df['readmitted']



    X_train, X_test, y_train, y_test = train_test_split(

        X, Y, test_size=0.3, stratify=Y, random_state=11)
```

```
    warnings.filterwarnings('ignore')


    numeric_transformer = RobustScaler(with_centering=False)
    categorical_transformer = OneHotEncoder(handle_unknown='ignore')
    preprocessor = ColumnTransformer(
        transformers=[
            ('cat', categorical_transformer, categorical_features),
            ('num', numeric_transformer, numeric_features)])


    stratified_kfold = StratifiedKFold(
        n_splits=10, shuffle=True, random_state=11)


    # General strengths range
    # param_grid = [{'classifier__C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],


    # Best Non-SMOTE
    # param_grid = [{'classifier__C': [0.09,
    #                                  0.091, 0.092, 0.093, 0.094, 0.095,
    #                                  0.096, 0.097, 0.098, 0.099,
    #                                  0.1,
    #                                  0.11, 0.12, 0.13, 0.14, 0.15,
    #                                  0.16, 0.17, 0.18, 0.19,
    #                                  0.2],


    # Best SMOTE
    param_grid = [{'classifier__C': [0.001,
```

```
                                        0.002, 0.003, 0.004, 0.005,
                                        0.006, 0.007, 0.008, 0.009,
                                        0.01,
                                        0.02, 0.03, 0.04, 0.05,
                                        0.06, 0.07, 0.08, 0.09,
                                        0.1],
                    'classifier__penalty': ['l1'],  # switch to L2 if want
                    'classifier__solver': ['saga']}]

    return (X_train, X_test, y_train, y_test,
            preprocessor, stratified_kfold, param_grid)


def find_score_NONSMOTE(X_train, X_test, y_train, y_test,
                        preprocessor, stratified_kfold, param_grid):
    pipeline = imbpipeline(steps=[
        ['preprocessor', preprocessor],
        ['classifier', LogisticRegression(
            random_state=11, max_iter=1000, n_jobs=-1)]])

    logreg = GridSearchCV(
        estimator=pipeline, param_grid=param_grid, scoring='roc_auc',
        cv=stratified_kfold, n_jobs=-1)

    logreg.fit(X_train, y_train)
    test_score = logreg.score(X_test, y_test)
```

```
    print(f'Non-SMOTE Cross-validation score: \t{logreg.best_score_}',
          f'\nNon-SMOTE Test score: \t\t{test_score}')

    print(logreg.best_params_)

def find_score_SMOTE(X_train, X_test, y_train, y_test,
                     preprocessor, stratified_kfold, param_grid):
    pipeline = imbpipeline(steps=[
        ['preprocessor', preprocessor],
        ['smote', SMOTE(random_state=11)],
        ['classifier', LogisticRegression(
            random_state=11, max_iter=1000, n_jobs=-1)]])

    logreg = GridSearchCV(
        estimator=pipeline, param_grid=param_grid, scoring='roc_auc',
        cv=stratified_kfold, n_jobs=-1)

    logreg.fit(X_train, y_train)
    test_score = logreg.score(X_test, y_test)

    print(f'SMOTE Cross-validation score: \t{logreg.best_score_}',
          f'\nSMOTE Test score: \t\t{test_score}')
```

```
    print(logreg.best_params_)


    threshold = 0.48
    y_pred = (
        logreg.predict_proba(X_test)[:, 1] > threshold).astype('float')
    print(classification_report(y_test, y_pred))
    ConfusionMatrixDisplay.from_predictions(
        y_test, y_pred, cmap='Blues', normalize='true')


    return logreg


def find_important(logreg, categorical_features, numeric_features):


    onehot_columns = list(
        logreg.best_estimator_.named_steps['preprocessor'].
        named_transformers_['cat'].
        get_feature_names(input_features=categorical_features))
    numeric_features_list = list(numeric_features)
    numeric_features_list.extend(onehot_columns)


    print(eli5.format_as_dataframe(eli5.explain_weights(
        logreg.best_estimator_.named_steps['classifier'],
        top=23, feature_names=numeric_features_list)).head(10))
```

```python
if __name__ == "__main__":
```

```python
    do_my_study()
```