EMORY UNIVERSITY

# Homework 3

Student name: *Lance Ding*

Course: *CS334* – Professor: *Dr. Shengpu Tang*
Due date: *10/2024*

### Feature Extraction

**(a) (i).** According to the documentation of *ICUType*, we have 4 possible values encoded numerically as such:

1: Coronary Care Unit
2: Cardiac Surgery Recovery Unit
3: Medical ICU
4: Surgical ICU

This use of numerical values to encode categorical variables in the setting of a linear classifier implies some sort of significant ordering of the different ICU types - that our variable is ordinal. This is problematic because our variable is actually nominal, and an encoding like this might imply something like the difference between a Cardiac Surgery Recovery Unit (encoded as a 2) and a Coronary Care Unit (encoded as a 1) is the same as the difference between a Surgical ICU (encoded as a 4) and a Medical ICU (encoded as a 3), when in reality the question doesn't even make any sense to begin with. A perhaps better way to do this is with a one-hot encoding, where each ICU type is represented with its own binary variable, where a 1 indicates that the patient was treated at that type of ICU and a 0 indicates otherwise.

**(a) (ii).** The usage of the mean for numerical variables might lead to misrepresentation due to outlier measurements, since the mean is not resistant to outliers. Additionally, the mean doesn't tell us anything about the spread of the data, which causes loss of information. Instead, we could consider something like the median, or even better, the median along with a metric for spread like an IQR or SD.

**(b) (i).** When we fill missing values with the means of that variable from all other patients who were not missing the value, we assuming that:

1. Misses are more or less random
2. Misses are independent
3. Data is not too skewed
4. All classes (in our case, different types of patients) tend to the same mean for that variable

There may be more assumptions not listed above, and these assumptions, when not met, can cause some issues for the conclusions that we want to draw. Instead, if the a variable is systematically missing data, we may consider removing the variable entirely from our analysis. Also, the fact that the value is missing may itself be utilized, so for each variable we can create a binary variable that indicates if the value was missing as a way of telling the model that there was some missing data.

**(c) (i).** It is useful to scale the features in this way so that when we train the model, it is less likely that we will run into numerical instability stemming from a large absolute difference of numbers, where precision would be lost. Also, this prevents features which are inherently larger from dominating the smaller features when norm-based optimization techniques are used.

**(d) (i).** The dimensionality $d$ of the feature vector is 40.

**(d) (ii).** The *average* feature vector, and corresponding feature names of the *X_train* returned by *get_train_test_split()*:

| | | |
|---|---|---|
| 0 | Age | 0.653507 |
| 1 | Gender | 0.557000 |
| 2 | Height | 0.390707 |
| 3 | ICUType | 0.578333 |
| 4 | Weight | 0.236098 |
| 5 | mean_ALP | 0.058284 |
| 6 | mean_ALT | 0.015867 |
| 7 | mean_AST | 0.013781 |
| 8 | mean_Albumin | 0.445900 |
| 9 | mean_BUN | 0.143020 |
| 10 | mean_Bilirubin | 0.039455 |
| 11 | mean_Cholesterol | 0.368926 |
| 12 | mean_Creatinine | 0.082277 |
| 13 | mean_DiasABP | 0.555640 |

| | | |
|---|---|---|
| 14 | mean_FiO2 | 0.359233 |
| 15 | mean_GCS | 0.720369 |
| 16 | mean_Glucose | 0.207822 |
| 17 | mean_HCO3 | 0.382337 |
| 18 | mean_HCT | 0.416416 |
| 19 | mean_HR | 0.467974 |
| 20 | mean_K | 0.338213 |
| 21 | mean_Lactate | 0.076291 |
| 22 | mean_MAP | 0.300090 |
| 23 | mean_Mg | 0.158574 |
| 24 | mean_NIDiasABP | 0.542114 |
| 25 | mean_NIMAP | 0.575976 |
| 26 | mean_NISysABP | 0.501373 |

| | | |
|---|---|---|
| 27 | mean_Na | 0.515676 |
| 28 | mean_PaCO2 | 0.322203 |
| 29 | mean_PaO2 | 0.284249 |
| 30 | mean_Platelets | 0.214156 |
| 31 | mean_RespRate | 0.350176 |
| 32 | mean_SaO2 | 0.937895 |
| 33 | mean_SysABP | 0.649529 |
| 34 | mean_Temp | 0.674652 |
| 35 | mean_TroponinI | 0.136583 |
| 36 | mean_TroponinT | 0.046039 |
| 37 | mean_Urine | 0.061263 |
| 38 | mean_WBC | 0.090518 |
| 39 | mean_pH | 0.011406 |

The average vector in *X_train*, values rounded to 4 decimal places.

**Hyperparameter Selection for L2-Regularized Logistic Regression**

**(a).** See code submission.

**(b).** It is beneficial to keep the class proportions roughly the same across folds to stop metrics from being misled. As we saw in class, certian metrics are sensitive to the proportion of one class to another such that we may get a high score on a metric when in reality our model could be terrible. By keeping the class proportions roughly the same across folds, we mitigate this problem.

**(c) (i).** See the table below:

| **Performance Measures** | **Best C** | **CV Performance** |
|---|---|---|
| Accuracy | 100 | 0.8665 |
| Precision | 1 | 0.6327 |
| Sensitivity | 100 | 0.2168 |
| Specificity | 0.001 | 1.0000 |
| F1-Score | 100 | 0.3171 |
| AUROC | 1 | 0.7982 |
| AUPRC | 10 | 0.4533 |

**(c) (ii).** As we varied *C*, we experienced different effects on the different metrics.

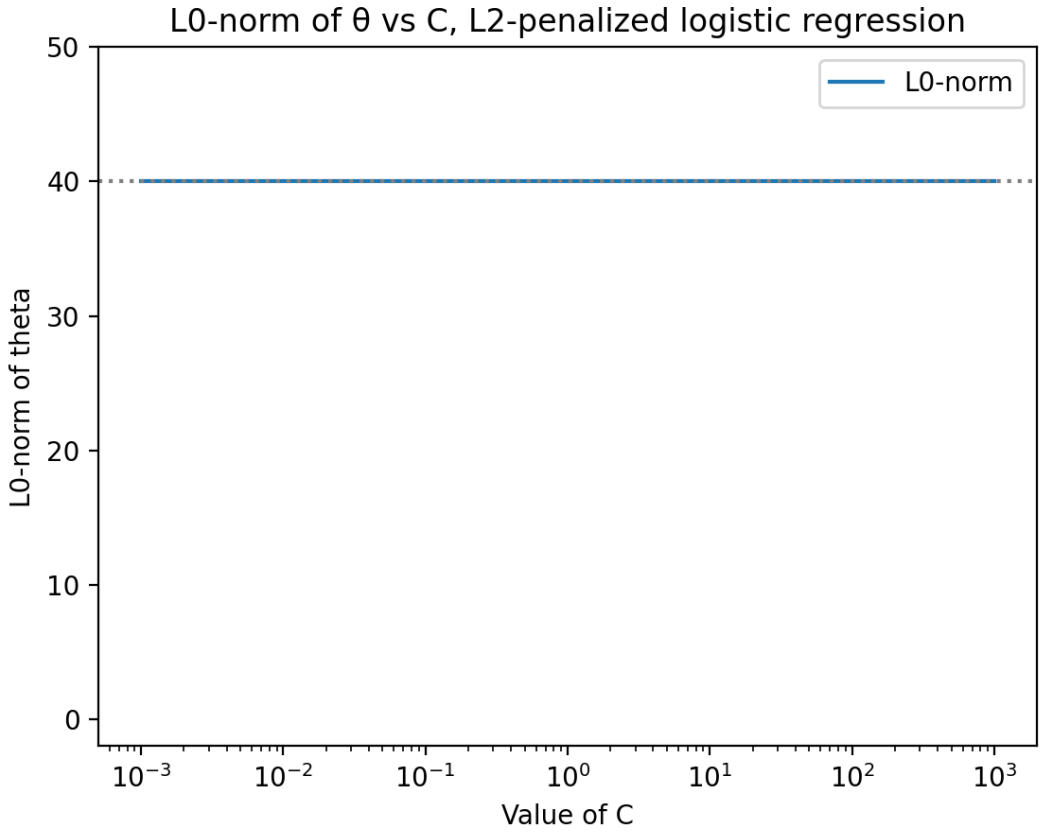
1. *Accuracy* did not show any significant sign of being affected by the different *C* values, only varying by about 0.1 through all the *C*s.
2. *Precision* saw big variations with different *C*s. With the lower *C*s, we saw a near-zero precision score, but with the higher *C*s the score rose to about 0.6
3. *Sensitivity* also saw variations with a similar pattern as *Precision* but on a smaller scale, starting at near-zero and rising to about 0.2 with the larger *C*s
4. *Specificity* did not see much variation, starting at 1 and ending at around 0.97 when going from the smallest to the largest *C*. It is interesting that we are able to get a specificity of 1 - this does not seem normal and should warrant caution and further investigation. My guess would be that the ratio of negative cases (patient did not die) to positive cases (patient died) is quite high, and the class imbalance causing this high score.
5. *F1-score* goes back to the pattern seen with *Precision* and *Sensitivity*, starting at near-zero and rising to about 0.3 with the higher *C*s
6. *AUROC* did not see much variation, staying between about 0.76 and 0.8 for all values of *C*.
7. *AUPRC* also did not see much variation, staying between about 0.42 and 0.45. I considered optimizing for this metric, since *AUPRC* is supposedly a good metric to use when we have potentially imbalanced class proportions, but because there was no significant improvement for the *AUPRC* by searching *C*, I opted to not.

Since we are dealing with life/death here, I believe it is important for us to try and optimize the ability of the model to actually predict that someone will die given that they did die - that is, we want to optimize the **sensitivity**, also known as the **recall**.

**(d).** According to our results, the best C then is 100.

| Chosen C = 100 | |
|---|---|
| **Performance Measures** | **CV Performance** |
| Accuracy | 0.8660 |
| Precision | 0.5758 |
| Sensitivity | 0.2639 |
| Specificity | 0.9673 |
| F1-Score | 0.3619 |
| AUROC | 0.8477 |
| AUPRC | 0.4825 |

**(e).** Below is the plot of the L0 norm of the weights VS different *C* values for the L2-penalized classifier.




We see that for all values of *C*, we do not get any 0 entries. This can be explained by the fact that L2 regularization does not tend to force feature weights to go to 0 - instead, it penalizes larger values and pushes the vector towards 0 but not exactly 0.

**(f).** See the tables below for the most positive and the most negative features, along with their weights we the classifier learned with L2 regularization and $C = 1$.

| **Positive Coefficient** | **Feature Name** |
|---|---|
| 2.8777 | mean_BUN |
| 2.1420 | Age |
| 1.6587 | mean_Bilirubin |
| 1.4451 | mean_Lactate |

| **Negative Coefficient** | **Feature Name** |
|---|---|
| -2.5321 | mean_GCS |
| -1.6250 | mean_Temp |
| -1.3351 | mean_Na |
| -0.9189 | Weight |

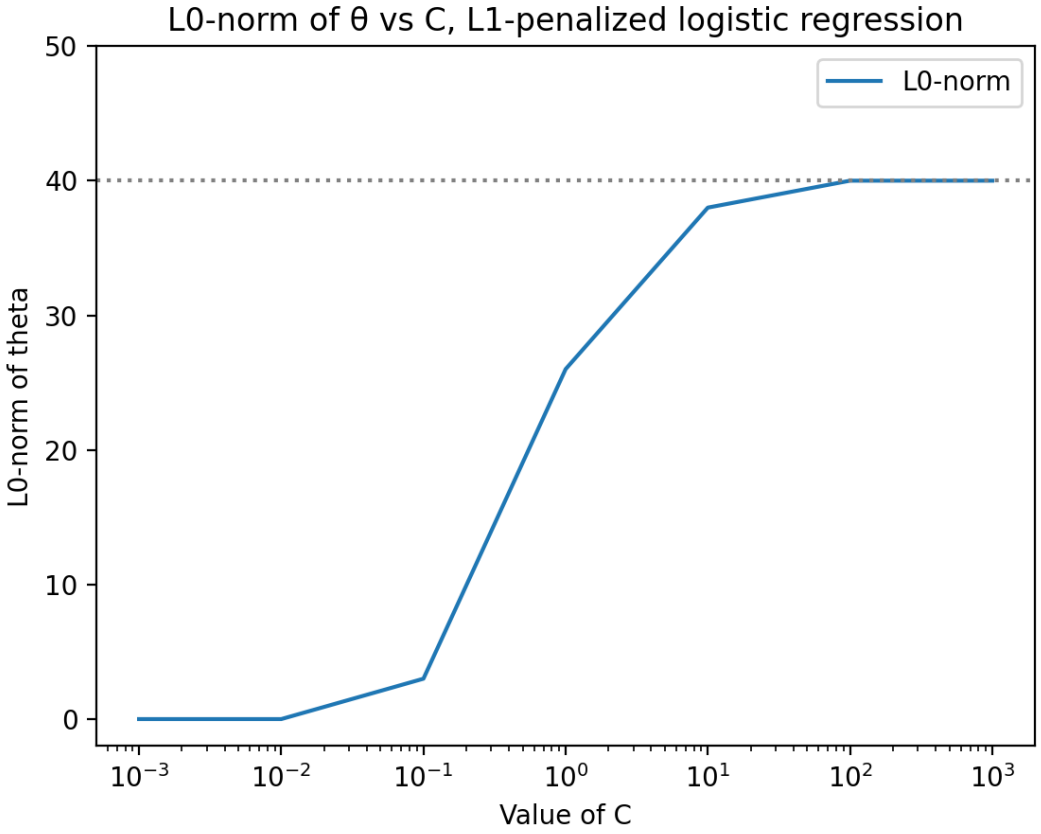
## Hyperparameter Selection for L1-Regularized Logistic Regression

**(a).** See code submission for details.
Best $C$ for AUROC from grid search: 1

| **Performance Measures** | **Best C** | **CV Performance** |
|---|---|---|
| Accuracy | 100 | 0.8665 |
| Precision | 1 | 0.6327 |
| Sensitivity | 100 | 0.2168 |
| Specificity | 0.001 | 1.0000 |
| F1-Score | 100 | 0.3171 |
| AUROC | 1 | 0.7982 |
| AUPRC | 10 | 0.4533 |

| Chosen C = 1 | |
|---|---|
| **Performance Measures** | **Test Performance** |
| Accuracy | 0.8660 |
| Precision | 0.5758 |
| Sensitivity | 0.2639 |
| Specificity | 0.9673 |
| F1-Score | 0.3619 |
| AUROC | 0.8480 |
| AUPRC | 0.4848 |

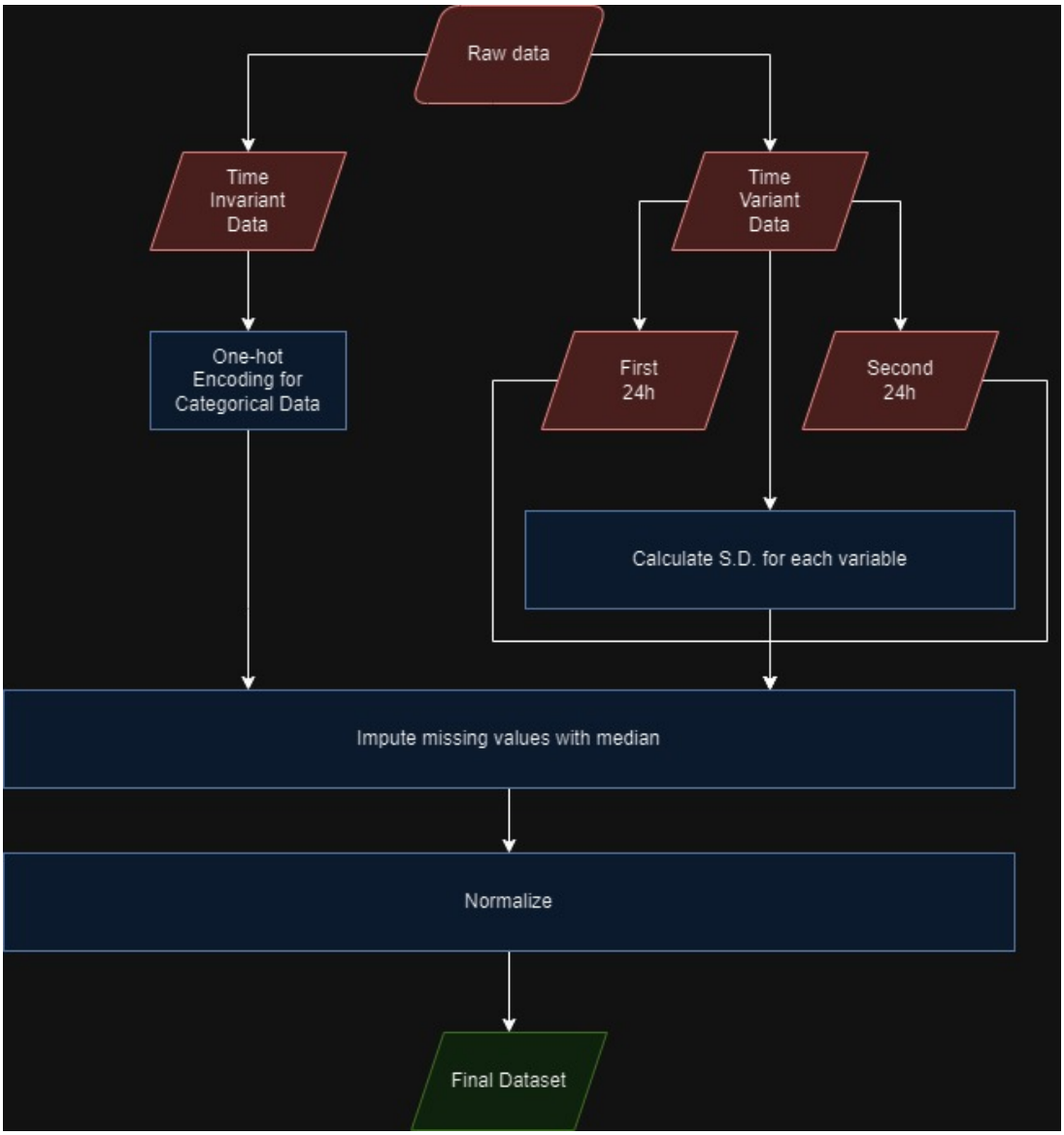
**(b).** Below is the plot of the L0 norm of weights VS different $C$ values for the L1-penalized classifier.



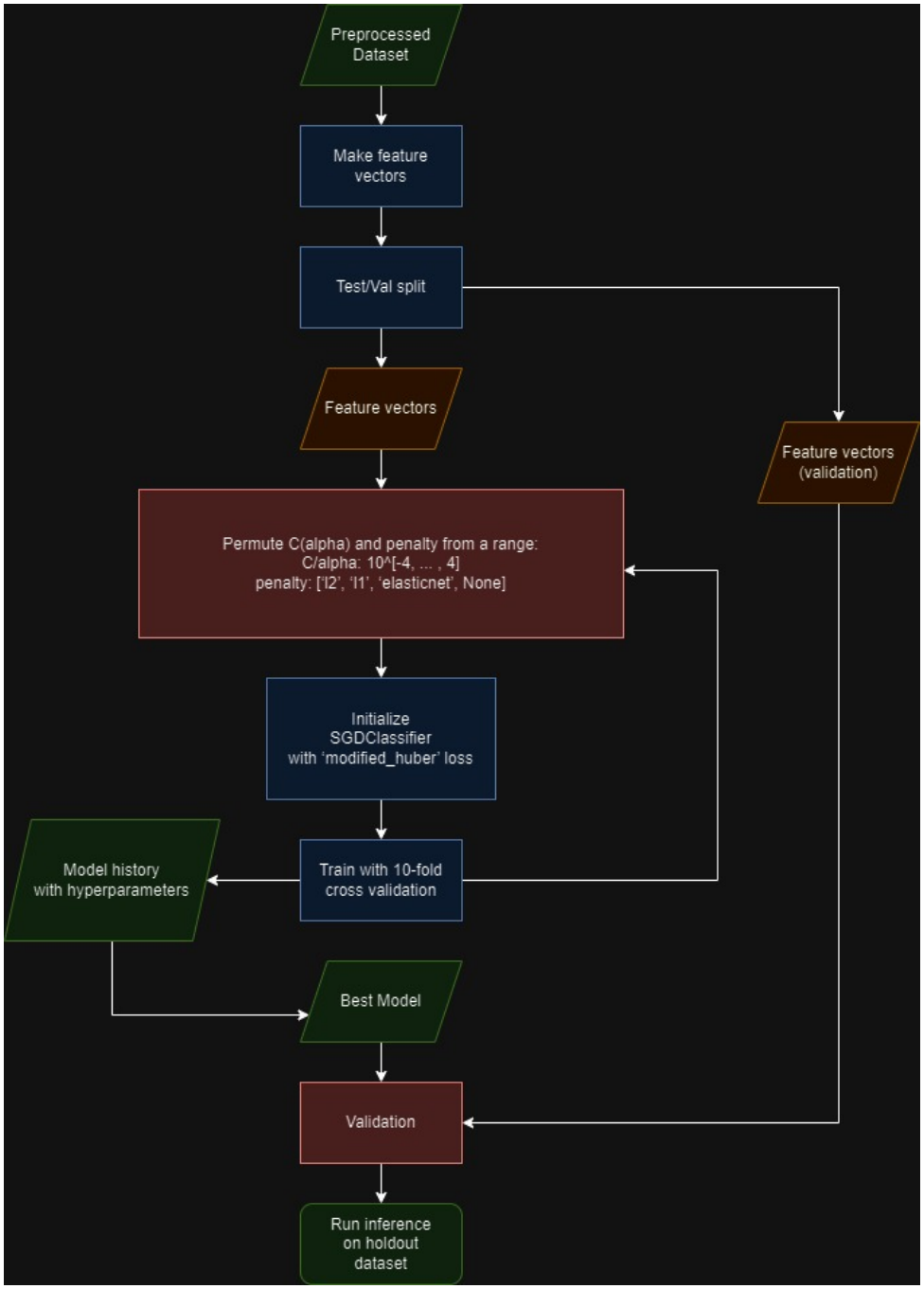

As we can see, unlike the graph for the L2-penalized classifier, there is activity here. At the smaller values of $C$, we observe smaller L0 norms, even actually reaching 0 at $C \leq 10^{-2}$. The L0 norm grows with $C$, until it hits 40, which is the length of our weights vector $\vec{\theta}$ at $C \geq 10^{2}$. This is probably due to the L1 penalty's tendency to reduce the model complexity by sparsifying features - that is, by applying L1 regularization with high strength, we expect to eventually see some coefficients become 0. Since $C$ is inversely proportional to the regularization strength $\lambda$, this is in line with what we would expect.

## Challenge

**Feature Engineering.** We elected to do feature engineering a little differently than in the methods above. Firstly, we deal with the categorical variables *Gender* and *ICUType* by using one-hot encoding. We also calculate the average values for time-varying data in the first 24h of the ICU stay and the second 24h of the ICU stay separately, and use them as features. A normalized standard distribution for time-variant data is also calculated over the entire 48 hour period. All missing values are then inputed with the median at this point, and normalized the same way as before. As a flowchart:



**Model Selection.** We decide to use a *SGDClassifier* with the *modified_huber* loss. We then conduct a grid search over the penalties: $l1, l2, elasticnet$ and the regularization strength constant $\alpha \in \{10^k \mid k \in \mathbb{Z}, -4 \leq k \leq 4\}$. We employ 10-fold cross validation during the hyperparamter selection process, and opted to optimize the F1 score. The best model was then selected, validated, and used in the final task. As a flowchart:



The model's validation performance on a holdout set that was 20% the size of the raw training dataset is as follows:

```
alpha = 0.0001
penalty = l1
Test Performance on metric f1_score: 0.4108
Validation Performance on metric accuracy: 0.8580
Validation Performance on metric precision: 0.5690
Validation Performance on metric sensitivity: 0.3214
Validation Performance on metric specificity: 0.9557
Validation Performance on metric f1_score: 0.4108
Validation Performance on metric auroc: 0.8208
Validation Performance on metric auprc: 0.4619
```

The confusion matrix on the training dataset is as such:

```
array([[0.88210121, 0.11789879],
       [0.32924962, 0.67075038]])
```