1(a)

i)

ICUType (1: Coronary Care Unit, 2: Cardiac Surgery Recovery Unit, 3: Medical ICU, or 4: Surgical ICU)

The current use of ICUType as a single numerical feature (1, 2, 3, 4) introduces artificial ordering relationships in the use of linear classifiers, suggesting some sort of "numerical size" or "order difference," but this is not practical because there is no size difference between these ICU types.

A better representation is to use one-hot Encoding, which converts the ICUType into four binary features, One for each ICU type, to eliminate the implicit effect of numerical order. For example, ICUType=2 becomes [0, 1, 0, 0].

ii)

Using the maximum value of a numerical variable as a feature representation has the following limitations: 1)Ignoring time change: Taking only the maximum value ignores the trend of the value changing over time. 2)Other important information is missing: the maximum value does not reflect the overall distribution of the variable. 3)Susceptibility to outliers: A high value for a single anomaly can greatly affect the maximum value, resulting in misleading features.

Other useful statistics: Mean, Median, Standard deviation, Minimum, Rate of change over time, etc.

1(b)

Assumption: This interpolation method assumes that the missing value is a random missing value (that is, the missing data is not biased toward some particular case), and that the mean value of the feature is a reasonable substitute value that is representative of the overall distribution.

Median interpolation: Replace missing values with median values to prevent the mean from being affected by extreme values; Regression interpolation: Use other features to predict missing values; Delete missing data: Directly delete the sample containing missing values, which is suitable for a small number of missing cases; Using KNN or other advanced interpolation methods: The missing value is estimated based on the characteristics of similar samples.

1(c)

i)

The goal of regularization is to control the complexity of the model by adding a penalty term to the loss function. For features that are not normalized, the range of eigenvalues varies greatly, which leads to uneven effects of regularization on different features. If a feature has a large numerical range (e.g. 1000 to 5000), then regularization may tend to penalize the weight of the feature when optimizing. Conversely, features with smaller values, such as 0 to 1, are less affected. Normalization ensures that the influence of each feature on regularization is balanced, and there is no bias due to differences in value ranges.

1(d)

i)

	Feature Name	Mean	IQR
0	Age	0.653843	0.341216
1	Gender	0.545347	1.000000
2	Height	0.394529	0.001172
3	ICUType	0.582917	0.666667
4	Weight	0.234589	0.127943
5	max_ALP	0.041011	0.011849
6	max_ALT	0.018456	0.016235
7	max_AST	0.017006	0.014794
8	max_Albumin	0.495221	0.000000
9	max_BUN	0.137952	0.108247
10	max_Bilirubin	0.046643	0.031569
11	max_Cholesterol	0.355707	0.000000
12	max_Creatinine	0.065124	0.037314
13	max_DiasABP	0.307588	0.038246
14	max_FiO2	0.764098	0.237453
15	max_GCS	0.878414	0.166667
16	max_Glucose	0.103543	0.060585
17	max_HCO3	0.355776	0.131579
18	max_HCT	0.296800	0.158442
19	max_HR	0.221652	0.112033
20	max_K	0.101920	0.034314
21	max_Lactate	0.093132	0.034507
22	max_MAP	0.387190	0.054435
23	max_MechVent	0.000000	0.000000
24	max_Mg	0.132897	0.045455
25	max_NIDiasABP	0.441452	0.122222
26	max_NIMAP	0.492288	0.105084
27	max_NISysABP	0.526196	0.105839
28	max_Na	0.375409	0.086207
29	max_PaCO2	0.376830	0.095238
30	max_PaO2	0.475380	0.302215
31	max_Platelets	0.204206	0.113725
32	max_RespRate	0.159428	0.000000
33	max_SaO2	0.913742	0.009530
34	max_SysABP	0.529797	0.067797
35	max_Temp	0.353835	0.138462
36	max_TroponinI	0.178032	0.000000
37	max_TroponinT	0.050695	0.000000
38	max_Urine	0.114734	0.076923
39	max_WBC	0.095309	0.048875
40	max_pH	0.004634	0.001485

2.1

(a)

Keeping class proportions (that is, the proportion of positive and negative samples) consistent across folds ensures that models are exposed to a similar class distribution when trained and validated. This helps to avoid the model being overly biased towards one class in some folds, thereby improving the stability and generalization of cross-validation results, especially when dealing with data that is categorically unbalanced.

(b)

In the optimization equation of logistic regression, the parameter C controls the tradeoff between the regularization penalty term J(θ) and the logical loss (i.e. the model's fit to the training data).

What C does: C is the reciprocal of the regularization strength. The smaller C puts more emphasis on regularization terms, simplifies the model, and avoids overfitting by punishing large coefficients in the logistic regression model. A larger C reduces the influence of the regularization term and allows the model to fit the training data more closely, capturing more complex relationships.

(c)

Forget to capitalize here lol…

Performance Measure	C	Penalty	Mean (Min, Max) CV Performance
accuracy	1	l2	0.8637 (0.8531, 0.8844)
precision	1	l2	0.5740 (0.4000, 0.9000)
f1-score	100	l2	0.2010 (0.0800, 0.3273)
auroc	1	l1	0.7982 (0.7563, 0.8462)
average_precision	1	l1	0.4380 (0.3809, 0.5890)
sensitivity	100	l2	0.1232 (0.0444, 0.2000)
specificity	0.001	l2	0.9862 (0.9781, 0.9964)

When selecting the final model, it is recommended to optimize the AUROC because it measures the ability of the model to distinguish between positive and negative classes at different thresholds, especially for imbalanced data sets. Compared to other measures, AUROC focuses more on the overall performance of the model rather than accuracy or precision under a single threshold, so it can better capture the classification ability of the model.

NOTE: I found that my f1-score and sensitivity performance have a “small” difference with the answers of my friends but the logic of our code is exactly the same, I guess there might be something wrong with my data itself but my code is most possibly correct.

(d)

The choice of C is 1, and penalty is l1.

Metric	Median	95% CI Lower	95% CI Upper
Accuracy	0.8675	0.8325	0.8975
Precision	0.6000	0.3333	0.8422
F1-score	0.2439	0.1081	0.3768
Auroc	0.8150	0.7463	0.8693
Average_precision	0.5019	0.3696	0.6285
Sensitivity	0.1552	0.0667	0.2593
Specificity	0.9827	0.9678	0.9942

(e)

From the L0-norm graph we generated, we can see the following trends:

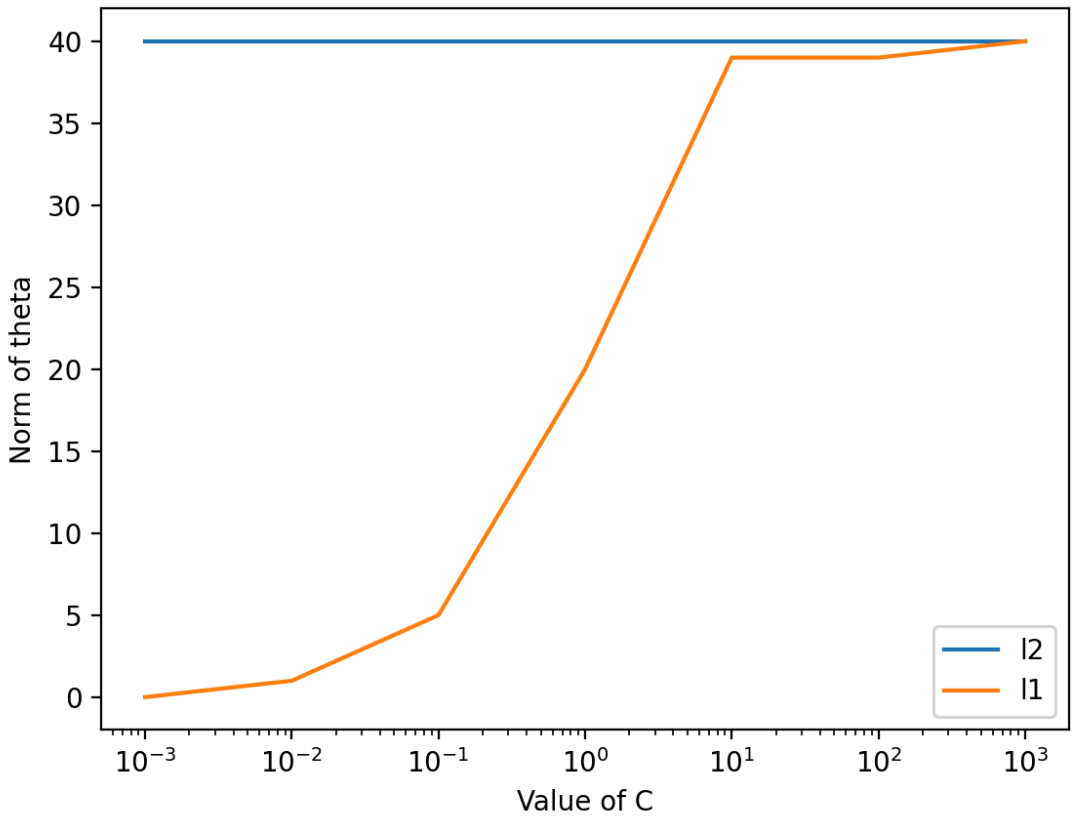
- L1 regularization (orange curve):

When C is small (C is close to 10^-3), the number of coefficients in the model is close to 0, which indicates that the model is very sparse and most coefficients are suppressed to 0.

With the increase of C, the L0-norm increases gradually, which indicates that the non-zero coefficients of the model increase gradually. The intensity of regularization decreases and the model becomes more complex. When C increases to a certain threshold (about 10^1), the L0-norm tends to stabilize, and the number of non-zero coefficients in the model no longer changes significantly.

- L2 regularization (blue curve):

Regardless of the size of the C value, the L0-norm of L2 regularization remains almost constant and close to the maximum. This shows that L2 regularization does not lead to sparsity as L1 does, it limits the size of the coefficient more than it compacting it to 0.



(f)

```
Top 4 Positive Coefficients and Corresponding Features:
         Feature  Coefficient
0    max_Lactate     3.558727
1        max_BUN     3.310016
2  max_Bilirubin     2.839005
3            Age     2.047908

Top 4 Negative Coefficients and Corresponding Features:
       Feature  Coefficient
0      max_GCS    -2.895462
1    max_Urine    -2.368379
2     max_Temp    -1.314032
3  max_DiasABP    -0.933855
```

From these positive and negative coefficients and features, most of the results make sense.

- Top Positive Coefficients:

max_Lactate and max_BUN: Elevated levels of lactic acid and blood urea nitrogen are often associated with physical stress or disease severity, especially organ failure or metabolic disorders, which may increase the risk of hospitalization death. (make sense)

max_Bilirubin: Elevated bilirubin may indicate liver function impairment or failure, which is also an indicator of a high risk of death. (make sense)

Age: The older the age, the higher the risk of death in hospital. (make sense)

- Top Negative Coefficients:

max_GCS: The higher the Glasgow Coma Score (GCS), the better the patient's state of consciousness and the lower the risk of death, this result is reasonable. (make sense)

max_Urine: Higher urine output usually indicates better kidney function and a lower risk of death. (make sense)

max_Temp: Higher body temperature may signal inflammation or infection, but in some cases may also be associated with lower mortality, which may depend on the specific pathological context. (-)

max_DiasABP: A higher diastolic blood pressure may indicate that the patient has better heart function and therefore a lower risk of death. (make sense)

3.1

(a)

The adjustment assigns distinct weights Wp and Wn to the positive and negative classes, respectively. These weights enable us to modify the impact of misclassifying positive or negative data during the optimization process.

If Wp significantly exceeds Wn, the model will prioritize the accurate classification of positive samples above that of negative ones. In other words, when $Wp \gg Wn$, the model is likely to predict a greater number of positive labels, even if this results in an elevated error rate for negative labels. This may result in increased Recall (Sensitivity) for positive instances but decreased Specificity, as the model becomes more permissive in misclassifying negative instances. In reality, this implies that the model will exert more effort to accurately forecast positive instances, perhaps resulting in the misclassification of negative instances.

(b)

Metric	Median	95% CI Lower	95% CI Upper
Accuracy	0.3125	0.2725	0.3600
Precision	0.1697	0.1307	0.2117
F1-score	0.2891	0.2299	0.3489
Auroc	0.8139	0.7595	0.8631
Average_precision	0.4794	0.3437	0.5943
Sensitivity	0.9836	0.9434	1.0000
Specificity	0.2046	0.1643	0.2463

(c)

- The biggest changes are:

Sensitivity: increased from 0.1552 to 0.9836. By focusing the weight on the positive class (class 1), the model is more inclined to correctly identify the positive class sample, so the Sensitivity is greatly improved.

Specificity: Specificity decreased from 0.9827 to 0.2046. Due to the low weight of negative class (-1 class), the model's ability to correctly classify negative class samples is significantly reduced.

- The least changes are:

AUROC and Average_precision changed little, with AUROC going from 0.815 to 0.8139, while Average_precision decreased from 0.5019 to 0.4794. These indicators comprehensively consider the performance of the model on two types of samples, rather than relying on a single type, so the impact of weight adjustment is less.

3.2

(a)

Strategy: 1) Initial Hypothesis: Given the dataset's imbalance with a scarcity of positive instances, we will assign a greater weight to the positive class (Wp) while maintaining the negative class weight (Wn) at about 1; 2) Cross Validation: Employ 5-fold cross-validation to evaluate various weight combinations for Wn and Wp. One may adjust Wp within the range of 5 to 100, whereas Wn remains constant at 1. This will enable you to assess the effects on performance metrics such as F1-score, AUROC, and Sensitivity; 3) Performance Metric: Prioritize the optimization of F1-score and AUROC, since they are effective indicators for addressing unbalanced data. The F1-score equilibrates precision and Recall (Sensitivity), whereas the AUROC provides a comprehensive assessment of the classifier's ability to distinguish between positive and negative classifications.

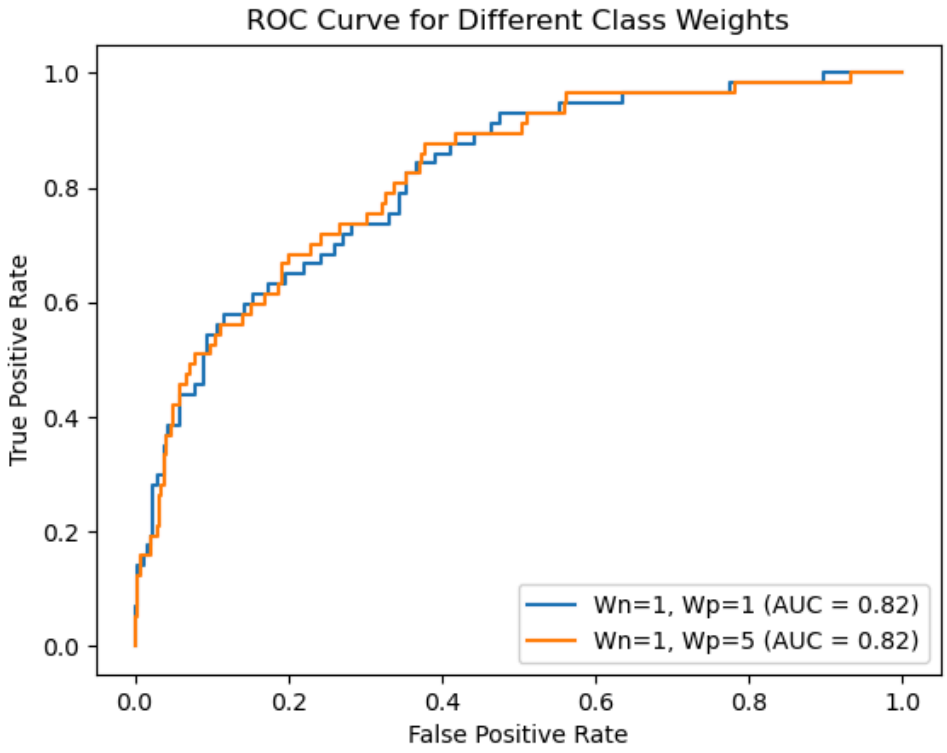
(b)

As mentioned in (a), we choose Wn = 1 and Wp = 40.

Metric	Median	95% CI Lower	95% CI Upper
Accuracy	0.3450	0.3025	0.3901
Precision	0.1745	0.1350	0.2169
F1-score	0.2936	0.2350	0.3553
Auroc	0.8138	0.7598	0.8633
Average_precision	0.4789	0.3449	0.5952
Sensitivity	0.9667	0.9138	1.0000
Specificity	0.2416	0.1965	0.2861

3.3

(a)



(b)

To enhance Sensitivity (Recall for the positive class) without altering class weights, we might:

1) Reduce the decision threshold used by the classifier for making predictions: Typically, several classifiers, such as logistic regression, assign class labels using a threshold of 0.5. Rather of using a threshold of 0.5 for predicting the positive class, consider reducing it (e.g., to 0.3). This indicates that a greater number of events will be categorized as positive, perhaps enhancing sensitivity.
2) Employ Threshold Tuning: Employ cross-validation on the training dataset to evaluate several thresholds and choose the one that optimally balances sensitivity with other metrics such as accuracy or specificity. This method operates without the need for model retraining and enables the adjustment of sensitivity for a pre-trained classifier.

4

(a)

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y^{(i)} - \theta^{T} \cdot x^{(i)})^{2} + \lambda \sum_{j=1}^{d} \theta_{j}^{2}$$

Differences:

Loss Function: Ridge regression minimizes squared error (regression task), while logistic regression minimizes binary cross-entropy loss (classification task).

Output Type: Ridge regression outputs continuous values (useful for regression tasks), whereas logistic regression outputs probabilities (used for binary classification).

(b)

Metric	Median	95% CI Lower	95% CI Upper
Accuracy	0.8675	0.8350	0.9025
Precision	0.6471	0.3747	0.9091
F1-score	0.2500	0.1132	0.3750
Auroc	0.8182	0.7586	0.8712
Average_precision	0.5050	0.3670	0.6219
Sensitivity	0.1538	0.0645	0.2593
Specificity	0.9855	0.9709	0.9971

Metric	Median	95% CI Lower	95% CI Upper
Accuracy	0.8725	0.8375	0.9050
Precision	0.7647	0.5000	1.0000
F1-score	0.2540	0.1270	0.4000
Auroc	0.5736	0.5312	0.6213
Average_precision	0.2393	0.1561	0.3307
Sensitivity	0.1559	0.0702	0.2581
Specificity	0.9913	0.9796	1.0000

The upper one is the performance of clf_Logistic and the lower one is the performance of clf_KernelRidge.

- The main differences include:

Average_precision: Logistic Regression has a higher Average_precision, which indicates that it has a better precision across different recall levels, reflecting stronger overall performance in ranking positive instances higher than negative ones.

Precision: Kernel Ridge Regression has a higher Precision score, this means that it is more accurate than Logistic Regression, indicating that Kernel Ridge Regression performs better on the accuracy of positive samples.

AUROC: Logistic Regression has a higher AUROC score, which indicates that it is better at distinguishing between positive and negative classes.

- Why the difference?

Logistic Regression is specifically designed for classification tasks and directly optimizes the classification's loss function, so it performs better on some classification tasks, such as AUROC. Kernel Ridge Regression is a Regression method, and although it can be used for classification tasks, it essentially optimizes continuously valued loss functions, so it may not be optimized specifically for classification problems as Logistic Regression is. This may result in poorer performance on AUROC than on Logistic Regression.

(d)

Gamma	Performance
0.001	0.5000 (0.5000, 0.5000)
0.010	0.5027 (0.4944, 0.5375)
0.100	0.5484 (0.4947, 0.6429)
1.000	0.5855 (0.5125, 0.6992)
10.000	0.5510 (0.4786, 0.6298)
100.000	0.5855 (0.4837, 0.7028)

For minimal γ values (e.g., 0.001 and 0.01), the performance remains consistently low and stable. This indicates that the model possesses restricted adaptability and fails to capture the fundamental non-linear patterns within the data.

As γ escalates (e.g., from 0.1 to 1.0), performance markedly enhances. An elevated γ enables the model to align more closely with the data, hence capturing more intricate correlations.

Beyond a specific threshold (γ = 10 or 100), the performance varies a lot, interestingly, when γ = 100, it nearly has the same performance as γ = 1, making the overall performance back to stable, while the mean performance remains the same.

(e)

C: 0.01, Gamma: 0.01
Accuracy – Median: 0.8575, 95% CI: (0.8225, 0.8925)
Precision – Median: 0.0000, 95% CI: (0.0000, 0.0000)
F1-score – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Auroc – Median: 0.5000, 95% CI: (0.5000, 0.5000)
Average_precision – Median: 0.1425, 95% CI: (0.1100, 0.1775)
Sensitivity – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)
C: 0.01, Gamma: 0.1
Accuracy – Median: 0.8575, 95% CI: (0.8225, 0.8925)
Precision – Median: 0.0000, 95% CI: (0.0000, 0.0000)
F1-score – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Auroc – Median: 0.5000, 95% CI: (0.5000, 0.5000)
Average_precision – Median: 0.1425, 95% CI: (0.1100, 0.1800)
Sensitivity – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)
C: 0.01, Gamma: 1
Accuracy – Median: 0.8575, 95% CI: (0.8225, 0.8925)
Precision – Median: 0.0000, 95% CI: (0.0000, 0.0000)
F1-score – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Auroc – Median: 0.5000, 95% CI: (0.5000, 0.5000)
Average_precision – Median: 0.1425, 95% CI: (0.1075, 0.1751)
Sensitivity – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)
C: 0.01, Gamma: 10
Accuracy – Median: 0.8750, 95% CI: (0.8425, 0.9050)
Precision – Median: 1.0000, 95% CI: (1.0000, 1.0000)
F1-score – Median: 0.2143, 95% CI: (0.0845, 0.3544)
Auroc – Median: 0.5599, 95% CI: (0.5250, 0.6035)
Average_precision – Median: 0.2473, 95% CI: (0.1716, 0.3281)
Sensitivity – Median: 0.1203, 95% CI: (0.0425, 0.2076)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)

C: 0.1, Gamma: 0.01
Accuracy – Median: 0.8575, 95% CI: (0.8225, 0.8925)
Precision – Median: 0.0000, 95% CI: (0.0000, 0.0000)
F1-score – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Auroc – Median: 0.5000, 95% CI: (0.5000, 0.5000)
Average_precision – Median: 0.1425, 95% CI: (0.1100, 0.1775)
Sensitivity – Median: 0.0000, 95% CI: (0.0000, 0.0000)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)
C: 0.1, Gamma: 0.1
Accuracy – Median: 0.8650, 95% CI: (0.8300, 0.8975)
Precision – Median: 1.0000, 95% CI: (0.0000, 1.0000)
F1-score – Median: 0.0968, 95% CI: (0.0000, 0.2105)
Auroc – Median: 0.5263, 95% CI: (0.5000, 0.5593)
Average_precision – Median: 0.1876, 95% CI: (0.1282, 0.2498)
Sensitivity – Median: 0.0508, 95% CI: (0.0000, 0.1177)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)
C: 0.1, Gamma: 1
Accuracy – Median: 0.8700, 95% CI: (0.8350, 0.9026)
Precision – Median: 0.7857, 95% CI: (0.4444, 1.0000)
F1-score – Median: 0.2059, 95% CI: (0.0789, 0.3377)
Auroc – Median: 0.5582, 95% CI: (0.5186, 0.6053)
Average_precision – Median: 0.2218, 95% CI: (0.1441, 0.3126)
Sensitivity – Median: 0.1214, 95% CI: (0.0468, 0.2143)
Specificity – Median: 0.9942, 95% CI: (0.9828, 1.0000)
C: 0.1, Gamma: 10
Accuracy – Median: 0.8675, 95% CI: (0.8375, 0.9000)
Precision – Median: 0.7778, 95% CI: (0.5000, 1.0000)
F1-score – Median: 0.2029, 95% CI: (0.0822, 0.3381)
Auroc – Median: 0.5571, 95% CI: (0.5175, 0.6066)
Average_precision – Median: 0.2180, 95% CI: (0.1437, 0.3055)
Sensitivity – Median: 0.1194, 95% CI: (0.0454, 0.2106)
Specificity – Median: 0.9942, 95% CI: (0.9855, 1.0000)

C: 1.0, Gamma: 0.01
Accuracy – Median: 0.8650, 95% CI: (0.8299, 0.9025)
Precision – Median: 1.0000, 95% CI: (0.0000, 1.0000)
F1-score – Median: 0.0968, 95% CI: (0.0000, 0.2090)
Auroc – Median: 0.5246, 95% CI: (0.5000, 0.5556)
Average_precision – Median: 0.1876, 95% CI: (0.1325, 0.2498)
Sensitivity – Median: 0.0511, 95% CI: (0.0000, 0.1229)
Specificity – Median: 1.0000, 95% CI: (1.0000, 1.0000)
C: 1.0, Gamma: 0.1
Accuracy – Median: 0.8725, 95% CI: (0.8375, 0.9050)
Precision – Median: 0.7647, 95% CI: (0.4545, 1.0000)
F1-score – Median: 0.2540, 95% CI: (0.1219, 0.3929)
Auroc – Median: 0.5733, 95% CI: (0.5299, 0.6246)
Average_precision – Median: 0.2367, 95% CI: (0.1582, 0.3266)
Sensitivity – Median: 0.1538, 95% CI: (0.0741, 0.2609)
Specificity – Median: 0.9913, 95% CI: (0.9800, 1.0000)
C: 1.0, Gamma: 1
Accuracy – Median: 0.8850, 95% CI: (0.8525, 0.9175)
Precision – Median: 0.7241, 95% CI: (0.5217, 0.8889)
F1-score – Median: 0.4348, 95% CI: (0.2898, 0.5625)
Auroc – Median: 0.6473, 95% CI: (0.5878, 0.7126)
Average_precision – Median: 0.3283, 95% CI: (0.2178, 0.4462)
Sensitivity – Median: 0.3148, 95% CI: (0.1875, 0.4464)
Specificity – Median: 0.9797, 95% CI: (0.9632, 0.9941)
C: 1.0, Gamma: 10
Accuracy – Median: 0.8775, 95% CI: (0.8450, 0.9100)
Precision – Median: 0.7500, 95% CI: (0.5000, 1.0000)
F1-score – Median: 0.3253, 95% CI: (0.1904, 0.4675)
Auroc – Median: 0.5973, 95% CI: (0.5508, 0.6538)
Average_precision – Median: 0.2731, 95% CI: (0.1777, 0.3743)
Sensitivity – Median: 0.2090, 95% CI: (0.1153, 0.3276)
Specificity – Median: 0.9884, 95% CI: (0.9763, 0.9971)

C: 10, Gamma: 0.01
Accuracy – Median: 0.8750, 95% CI: (0.8450, 0.9075)
Precision – Median: 0.8333, 95% CI: (0.5714, 1.0000)
F1-score – Median: 0.2571, 95% CI: (0.1200, 0.4000)
Auroc – Median: 0.5756, 95% CI: (0.5341, 0.6267)
Average_precision – Median: 0.2492, 95% CI: (0.1596, 0.3447)
Sensitivity – Median: 0.1579, 95% CI: (0.0678, 0.2600)
Specificity – Median: 0.9942, 95% CI: (0.9852, 1.0000)
C: 10, Gamma: 0.1
Accuracy – Median: 0.8750, 95% CI: (0.8400, 0.9075)
Precision – Median: 0.6500, 95% CI: (0.4499, 0.8462)
F1-score – Median: 0.3721, 95% CI: (0.2466, 0.5000)
Auroc – Median: 0.6201, 95% CI: (0.5655, 0.6782)
Average_precision – Median: 0.2775, 95% CI: (0.1813, 0.3929)
Sensitivity – Median: 0.2635, 95% CI: (0.1500, 0.3774)
Specificity – Median: 0.9768, 95% CI: (0.9578, 0.9911)
C: 10, Gamma: 1
Accuracy – Median: 0.8650, 95% CI: (0.8325, 0.8975)
Precision – Median: 0.5480, 95% CI: (0.3666, 0.7308)
F1-score – Median: 0.4000, 95% CI: (0.2727, 0.5208)
Auroc – Median: 0.6362, 95% CI: (0.5729, 0.6974)
Average_precision – Median: 0.2693, 95% CI: (0.1823, 0.3734)
Sensitivity – Median: 0.3204, 95% CI: (0.1896, 0.4427)
Specificity – Median: 0.9566, 95% CI: (0.9345, 0.9773)
C: 10, Gamma: 10
Accuracy – Median: 0.8725, 95% CI: (0.8400, 0.9025)
Precision – Median: 0.6400, 95% CI: (0.4209, 0.8261)
F1-score – Median: 0.3529, 95% CI: (0.2029, 0.4889)
Auroc – Median: 0.6103, 95% CI: (0.5567, 0.6707)
Average_precision – Median: 0.2686, 95% CI: (0.1772, 0.3763)
Sensitivity – Median: 0.2456, 95% CI: (0.1379, 0.3594)
Specificity – Median: 0.9768, 95% CI: (0.9585, 0.9912)

```
C: 100, Gamma: 0.01
Accuracy - Median: 0.8675, 95% CI: (0.8350, 0.9025)
Precision - Median: 0.6500, 95% CI: (0.3846, 0.8889)
F1-score - Median: 0.2532, 95% CI: (0.1250, 0.3896)
Auroc - Median: 0.5700, 95% CI: (0.5270, 0.6185)
Average_precision - Median: 0.2251, 95% CI: (0.1492, 0.3151)
Sensitivity - Median: 0.1552, 95% CI: (0.0667, 0.2554)
Specificity - Median: 0.9855, 95% CI: (0.9731, 0.9971)
C: 100, Gamma: 0.1
Accuracy - Median: 0.8750, 95% CI: (0.8400, 0.9050)
Precision - Median: 0.5946, 95% CI: (0.4074, 0.7778)
F1-score - Median: 0.4091, 95% CI: (0.2716, 0.5253)
Auroc - Median: 0.6405, 95% CI: (0.5773, 0.7028)
Average_precision - Median: 0.2860, 95% CI: (0.1910, 0.4023)
Sensitivity - Median: 0.3167, 95% CI: (0.1929, 0.4468)
Specificity - Median: 0.9654, 95% CI: (0.9449, 0.9827)
C: 100, Gamma: 1
Accuracy - Median: 0.8625, 95% CI: (0.8300, 0.8975)
Precision - Median: 0.5263, 95% CI: (0.3721, 0.6897)
F1-score - Median: 0.4138, 95% CI: (0.2926, 0.5410)
Auroc - Median: 0.6520, 95% CI: (0.5887, 0.7136)
Average_precision - Median: 0.2769, 95% CI: (0.1896, 0.3875)
Sensitivity - Median: 0.3519, 95% CI: (0.2407, 0.4844)
Specificity - Median: 0.9477, 95% CI: (0.9229, 0.9701)
C: 100, Gamma: 10
Accuracy - Median: 0.8725, 95% CI: (0.8400, 0.9026)
Precision - Median: 0.6250, 95% CI: (0.4286, 0.8000)
F1-score - Median: 0.3684, 95% CI: (0.2333, 0.5000)
Auroc - Median: 0.6179, 95% CI: (0.5638, 0.6800)
Average_precision - Median: 0.2740, 95% CI: (0.1838, 0.3844)
Sensitivity - Median: 0.2600, 95% CI: (0.1500, 0.3750)
Specificity - Median: 0.9739, 95% CI: (0.9558, 0.9885)
```

We know that we have our `Best AUROC achieved with C=100, Gamma=1, AUROC=0.6520`

```
C: 100, Gamma: 1
Accuracy - Median: 0.8625, 95% CI: (0.8300, 0.8975)
Precision - Median: 0.5263, 95% CI: (0.3721, 0.6897)
F1-score - Median: 0.4138, 95% CI: (0.2926, 0.5410)
Auroc - Median: 0.6520, 95% CI: (0.5887, 0.7136)
Average_precision - Median: 0.2769, 95% CI: (0.1896, 0.3875)
Sensitivity - Median: 0.3519, 95% CI: (0.2407, 0.4844)
Specificity - Median: 0.9477, 95% CI: (0.9229, 0.9701)
```

and its performance is .

(f)

In nonlinear models (such as Kernel Ridge Regression using RBF cores), it is not possible to directly report the weight of each feature because the kernel method works differently than the linear model. It is based on the similarity between the training samples (the similarity calculated by the kernel function). Therefore, the output (decision boundary) of the model is based on a combination of all the support vectors in a high-dimensional space, rather than directly from the input features, so the coefficients (or weights) of the model do not directly correspond to the features in the original feature space.

5

In this project, I applied several feature engineering techniques to handle different types of data:

1. Feature Engineering

- Numerical Features: The numerical features in the dataset were preprocessed by filling missing values with the median of each feature (imputation with the median strategy). After that, I scaled the numerical features using StandardScaler, which standardizes the data by removing the mean and scaling to unit variance. This is crucial for models like logistic regression, which are sensitive to feature scaling.

- Categorical Features: The categorical variables, such as Gender and ICUType, were processed using one-hot encoding. Missing values in categorical features were imputed using the most frequent category.

2. Hyperparameter Selection

For hyperparameter selection, I performed a **grid search** using cross-validation to identify the best regularization strength (C) for the logistic regression model. The grid search explored a range of C values: [0.01, 0.1, 1.0, 10, 100] with the l2 regularization penalty.

The evaluation metric for selecting the best parameters was AUROC, which is particularly suitable for imbalanced datasets. After performing cross-validation on the training data, the best C was found to be 0.1.

```
Best parameters: {'C': 0.1, 'penalty': 'l2'}
```

3. Model Training

I used a logistic regression model with the l2 penalty and the optimal C value identified from the grid search (C=0.1). The model was trained on the resampled training data, which was processed using SMOTE to handle the class imbalance in the dataset, which generates synthetic samples for the minority class to balance the dataset before training.

4. Held-Out Set Prediction

For the held-out test set, I processed it using the same preprocessing pipeline (including imputation, scaling, and one-hot encoding) as applied to the training set. I then used the trained logistic regression model to predict both binary labels (y_label) and risk scores (y_score) on the held-out data. The binary predictions were obtained using clf.predict(), and the risk scores were obtained using clf.decision_function().

5. Output Submission

The predictions for both y_label (binary predictions) and y_score (risk scores) were saved into a CSV file using the provided generate_challenge_labels() function. This function ensures the output file is in the correct format for submission.

6. Confusion Matrix

Finally, I calculated the confusion matrix using the true labels (y_test_subset) and predicted labels (y_cm) for the held-out set to evaluate the model's performance.

```
[[1909  637]
[ 131  323]]
```

7. Results

The final performance of the model was evaluated using multiple metrics, including AUROC, F1-score, accuracy, precision, sensitivity, and specificity. The confusion matrix provides a breakdown of true positive, true negative, false positive, and false negative rates, offering insight into the model's classification performance.