
Exploiting Negative Samples: A Catalyst for Cohort Discovery in Healthcare Analytics

Appendices

1 A Notation Table

2 In this paper, scalars are denoted by symbols such as x , vectors are represented by boldface symbols
3 such as \mathbf{x} , and matrices are described by uppercase boldface symbols such as \mathbf{X} . To provide a
4 comprehensive overview of the notations used throughout the paper, we present a summary of
5 notations in Table 1.

Table 1: Notations

Notation	Description
\mathcal{D}, d_i	EMR data, each sample in EMR data
$\mathcal{D}^+, \mathcal{D}^-$	Positive samples, negative samples
d_i^-	Each negative sample
\mathbf{x}_i, y_i	Input features of d_i^- , binary label of d_i^-
F	Prediction model
\mathcal{Q}	A subset of negative samples
M	Performance metric function
s_i	Data Shapley value for d_i^-
π	A Monte Carlo permutation
$A_{\pi}^{d_i^-}$	All the negative samples before d_i^- in π
\mathcal{S}	The Negative Sample Shapley Field
\mathcal{S}'	Transformed space after SDAE-based manifold learning
K	Number of DAEs in SDAE
k	Each DAE in SDAE, $k \in \{0, \dots, K-1\}$
$\mathbf{h}_i^{(k)}$	Input to the encoder of the k -th DAE
$\tilde{\mathbf{h}}_i^{(k)}$	Corrupted version of $\mathbf{h}_i^{(k)}$ with masking noise
$f_{\theta}^{(k+1)}(\cdot)$	Encoder of the k -th DAE
$\hat{\mathbf{h}}_i^{(k+1)}$	Output from the encoder of the k -th DAE
$f_{\phi}^{(k+1)}(\cdot)$	Decoder of the k -th DAE
$\mathbf{z}_i^{(k)}$	Output from the decoder of the k -th DAE
$\mathcal{L}_{rec}^{(k)}$	Reconstruction loss in the k -th DAE
$\mathcal{L}_{iso}^{(k)}$	Isotropy constraint in the k -th DAE
$\mathcal{L}^{(k)}$	Overall loss in the k -th DAE
$\mathbf{h}_i^{(k+1)}$	Input to the encoder of the $(k+1)$ -th DAE
$\mathbf{h}_i^{(K)}$	Input for medical cohort discovery

B Negative Sample Shapley Field Construction

B.1 Proof of Data Shapley Value for Negative Samples

We establish the proof of the data Shapley value for negative samples by relating our problem to the original context of the Shapley value in game theory [40], i.e., reducing it to a cooperative game [38, 12, 7].

Specifically, our problem is framed as a negative sample valuation game for a fair distribution of the collective performance achieved by the prediction model to each participating negative sample in the training data (with positive samples the same), while maintaining consistency with the three fundamental properties of an equitable data valuation: (i) null player, (ii) symmetry, and (iii) linearity.

Null player. We define a negative sample d_i^- as a “null player” and set its data Shapley value to zero if its inclusion in any subsets of the negative sample set in training data does not influence the performance of the prediction model. Formally, for a negative sample $d_i^- = (\mathbf{x}_i, y_i)$ and $\forall \mathcal{R} \subseteq \mathcal{D}^- \setminus d_i^-$, if the performance remains unchanged by adding d_i^- , i.e., $M(\mathcal{D}^+ \cup \mathcal{R}, F) = M(\mathcal{D}^+ \cup \mathcal{R} \cup \{d_i^-\}, F)$, then $s_i = 0$. In this negative sample valuation game, the null player property ensures that the negative samples with no impact on the prediction performance are assigned zero values for their data Shapley values.

Symmetry. Two negative samples, d_i^- , and d_j^- , are assigned the same value if they consistently influence the performance of the prediction model when added to any subsets of the negative sample set in training data. This property arises from the concept of symmetry. Formally, for two negative samples $d_i^- = (\mathbf{x}_i, y_i)$ and $d_j^- = (\mathbf{x}_j, y_j)$, and $\forall \mathcal{R} \subseteq \mathcal{D}^- \setminus \{d_i^-, d_j^-\}$, if the prediction performance remains the same after adding d_i^- or d_j^- , i.e., $M(\mathcal{D}^+ \cup \mathcal{R} \cup \{d_i^-\}, F) = M(\mathcal{D}^+ \cup \mathcal{R} \cup \{d_j^-\}, F)$, then $s_i = s_j$. This property ensures that the negative samples with equivalent marginal contributions are assigned the same data Shapley values.

Linearity. The influence of a negative sample d_i^- on the overall pooled data is equivalent to its influence on constituent sub-datasets. We could denote s_i as $s_i(d_i^-, \mathcal{D}_{test})$, representing the data Shapley value of the negative sample d_i^- evaluated on all test data \mathcal{D}_{test} . The linearity property states that for two sets of test data, \mathcal{D}_{test}^1 and \mathcal{D}_{test}^2 , the following holds:

$$s_i(d_i^-, \mathcal{D}_{test}^1 \cup \mathcal{D}_{test}^2) = s_i(d_i^-, \mathcal{D}_{test}^1) + s_i(d_i^-, \mathcal{D}_{test}^2) \quad (1)$$

This linearity property ensures that the data Shapley value of a negative sample on the pooled test dataset is equal to the sum of its data Shapley values on the two individual test datasets, in this negative sample valuation game.

Proposition 1 The data Shapley value s_i for a negative sample d_i^- is given by:

$$s_i = H \sum_{\mathcal{Q} \subseteq \mathcal{D}^- - \{d_i^-\}} \frac{M(\mathcal{D}^+ \cup \mathcal{Q} \cup \{d_i^-\}, F) - M(\mathcal{D}^+ \cup \mathcal{Q}, F)}{\binom{N^- - 1}{|\mathcal{Q}|}} \quad (2)$$

where H is a constant and the summation is taken over all subsets of negative samples, except d_i^- .

Proof 1 We prove Proposition 1 above by establishing the connection between our negative sample valuation game and the cooperative game theory context [7]. In a cooperative game, there exists a set of n players and a characteristic function $m : 2^{[n]} \mapsto \mathbb{R}$ that assigns a payment value to each selected player [38]. In our case, the players correspond to individual negative samples, and the characteristic function $m(\mathcal{Q})$ represents the performance obtained when the subset of negative samples \mathcal{Q} ($\mathcal{Q} \subseteq \mathcal{D}^-$) is included in the prediction model. By leveraging the three properties discussed above, our negative sample valuation game ensures the fair distribution of collective performance to the participating negative samples. Therefore, each negative sample acts as a player, and the prediction model F incorporates all the participating negative samples \mathcal{Q} (along with the positive samples \mathcal{D}^+) to achieve the overall performance $m = M(\mathcal{D}^+ \cup \mathcal{Q}, F)$. Consequently, the data Shapley value of each negative sample corresponds to the payment received by each player in this cooperative game analogy. \square

B.2 Monte Carlo Permutation Sampling

We adopt Monte Carlo permutation sampling to approximate the data Shapley values for negative samples. The detailed procedure of each Monte Carlo iteration is presented in Algorithm 1. The

53 algorithm begins by initializing the necessary variables for computation in lines 1-8. Subsequently,
 54 for a given permutation, we calculate the marginal contribution of each negative sample in the current
 55 Monte Carlo iteration towards its overall data Shapley value, as described in lines 9-25.

56 In particular, for each indexed negative sample, we include it in the training data and retrain the
 57 classifier. Then, we measure its marginal contribution by calculating the difference in the AUC metric
 58 (lines 10-14). Additionally, in line 15, we compute the absolute difference between the full AUC
 59 (which uses all the training data and evaluates the trained model on the test data) and the new AUC
 60 (which includes the current negative sample). If this difference falls below a predefined threshold,
 61 specifically “*truncation_tolerance*” times the full AUC, for more than five consecutive negative
 62 samples, we terminate the current Monte Carlo iteration by early stopping (lines 16-21). This early
 63 stopping criterion is based on the observation that further inclusion of negative samples is unlikely to
 64 yield a significant improvement in AUC.

65 After calculating the marginal contribution of each negative sample in each Monte Carlo iteration,
 66 the overall data Shapley value of a particular negative sample is derived by taking the mean of its
 67 marginal contributions across different iterations.

Algorithm 1: Data Shapley Value Computation for Negative Samples by Monte Carlo Sampling

Input : Negative training data $(\mathbf{X}_{train}^-, \mathbf{y}_{train}^-)$, Positive training data $(\mathbf{X}_{train}^+, \mathbf{y}_{train}^+)$, Test data $(\mathbf{X}_{test}, \mathbf{y}_{test})$.

Output : The marginal contribution of each negative sample in the current Monte Carlo iteration to its overall data Shapley value.

```

1 Initialize permutation of indices of  $\mathbf{X}_{train}^-$ :  $perm \leftarrow$  random permutation
2 Initialize marginal contributions of  $\mathbf{X}_{train}^-$  with zeros:  $marginal\_contributes \leftarrow$  zeros
3 Initialize truncation counter:  $truncation\_counter \leftarrow 0$ 
4 Initialize new score with a random score:  $new\_score \leftarrow$  random_score // 0.5 for AUC
5 Initialize a classifier:  $clf \leftarrow$  create a new classifier
6 Fit the classifier with all training data:  $clf.fit(\mathbf{X}_{train}^- \cup \mathbf{X}_{train}^+, \mathbf{y}_{train}^- \cup \mathbf{y}_{train}^+)$ 
7 Evaluate the classifier on test data:  $full\_score \leftarrow AUC(clf, \mathbf{X}_{test}, \mathbf{y}_{test})$ 
8 Initialize training data:  $(\mathbf{X}', \mathbf{y}') \leftarrow (\mathbf{X}_{train}^+, \mathbf{y}_{train}^+)$ 
9 for  $idx$  in  $perm$  do
10   Set old score to the current new score:  $old\_score \leftarrow new\_score$ 
11   Update training data with current negative sample:
68    $(\mathbf{X}', \mathbf{y}') \leftarrow (\mathbf{X}' \cup \mathbf{X}_{train}^-[idx], \mathbf{y}' \cup \mathbf{y}_{train}^-[idx])$ 
12   Create a new classifier and train it:  $clf \leftarrow$  new classifier,  $clf.fit(\mathbf{X}', \mathbf{y}')$ 
13   Update new score:  $new\_score \leftarrow AUC(clf, \mathbf{X}_{test}, \mathbf{y}_{test})$ 
14   Calculate the marginal contribution of the current negative sample:
       $marginal\_contributes[idx] \leftarrow new\_score - old\_score$ 
15   Calculate the distance to the full score:  $distance\_to\_full\_score \leftarrow |full\_score - new\_score|$ 
16   if  $distance\_to\_full\_score \leq truncation\_tolerance \times full\_score$  then
17     Increment truncation counter:  $truncation\_counter \leftarrow truncation\_counter + 1$ 
18     if  $truncation\_counter > 5$  then
19       break
20   end
21 end
22 else
23   Reset truncation counter:  $truncation\_counter \leftarrow 0$ 
24 end
25 end
26 return  $marginal\_contributes$ 

```

69 B.3 Implementation Details

70 In the experiments, we perform Monte Carlo permutation sampling 100,000 times to approximate the
 71 data Shapley values for negative samples. We adopt the early stopping criterion described earlier,
 72 where the threshold *truncation_tolerance* is set to 0.025. This means that if the absolute difference

73 between the full AUC and the new AUC falls within 0.025 times the full AUC for more than five
74 consecutive negative samples, the current Monte Carlo iteration will be terminated.

75 **C Manifold Learning with Structure Preservation and Isotropy Constraint**

76 **C.1 Implementation Details**

77 We utilize the Adam optimizer to train the SDAE in an unsupervised manner, using the loss
78 function described in Equation 7 of the paper. Our objective is to obtain an optimal manifold
79 space to support subsequent density-based clustering for automatic cohort identification. To deter-
80 mine the optimal learning rate for training, we perform a grid search over a range of values, i.e.,
81 $[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005]$, and run 10 repeats per learning rate. The model run with the
82 lowest loss is selected as the optimal model for subsequent cohort discovery, which corresponds to a
83 learning rate of 0.005. Other parameters are held constant during the training process, including a
84 batch size of 1024, a mask probability of 0.2 for the denoising process, and a total of 100 epochs.
85 These parameters provide stability and ensure sufficient training iterations to learn meaningful
86 representations in the SDAE.

87 **D Cohort Discovery Among High Data Shapley Value Negative Samples**

88 **D.1 Details for DBSCAN**

89 The DBSCAN (density-based spatial clustering of applications with noise) algorithm follows a
90 specific process to perform clustering. It requires two essential parameters: (i) ε , which defines the
91 maximum distance between two samples for them to be considered neighbors, and (ii) P_{min} , which
92 specifies the minimum number of samples required to form a dense region.

93 The detailed description of the DBSCAN algorithm is as follows. (i) Start by selecting an unvisited
94 sample arbitrarily. (ii) Retrieve its ε -neighborhood, consisting of all samples within a distance of
95 ε from the selected sample. (iii) If the ε -neighborhood contains more than P_{min} samples, initiate
96 a new cluster and designate the selected sample as a “core point”. The core point is a sample that
97 has a sufficient number of neighbors within its ε -neighborhood to form a dense region. (iv) If the
98 ε -neighborhood has fewer than P_{min} samples, label the selected sample as noise. However, note that
99 this sample may later fall within the ε -neighborhood of another sample, causing it to be assigned
100 to a different cluster. (v) For each sample that is determined to belong to a dense region within a
101 cluster, consider its ε -neighborhood as part of the same cluster. Add all the samples found within
102 this neighborhood to the cluster and check if these samples’ respective ε -neighborhoods are also
103 dense (if so, add them to the cluster as well). This process continues recursively until the entire
104 densely connected cluster is detected. (vi) Proceed to the next unvisited sample and repeat steps (ii)
105 to (v) until all samples have been assigned to a cluster or labeled as noise. By following this process,
106 DBSCAN identifies densely connected $\{C_1, C_2, \dots, C_R\}$ and recognizes noisy samples Ψ .

107 **D.2 Cluster vs. Cohort**

108 We illustrate the relationship between clusters and cohorts using an example within the DBSCAN
109 algorithm, as depicted in Figure 1. In this example, we set P_{min} to 4, and the value of ε is indicated
110 in the figure as the radius of the circles.

111 As shown in the figure, Point A and all the other blue points are core points because their ε -
112 neighborhoods contain at least P_{min} points. Therefore, they form a single cluster. Additionally, Point
113 B and Point C are reachable from Point A via existing paths, making them belong to the same cluster
114 as well. However, Point N is labeled as noise since it does not meet the criteria to be a core point and
115 is not reachable from any core points.

116 According to Definition 2 mentioned in the paper, for this identified cluster by the DBSCAN algorithm,
117 we consider each core point (i.e., all the blue points) and define a spherical space with the core point
118 as its center and ε as its radius. The combined area covered by all such spherical spaces, depicted in
119 blue, represents the cohort that we aim to discover from this cluster.

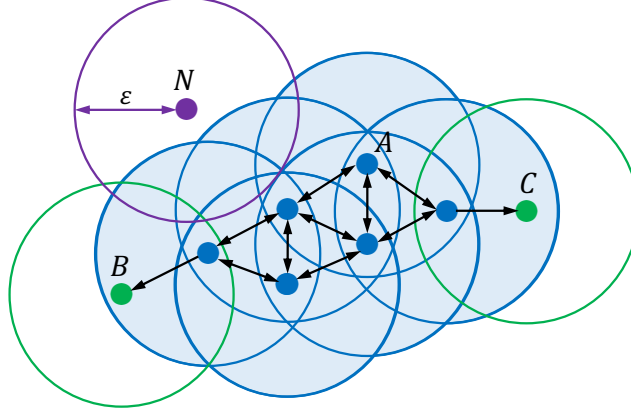


Figure 1: Relationship between clusters and cohorts in a DBSCAN example.

D.3 Implementation Details

The choice of parameters in the DBSCAN algorithm, specifically the search radius (ϵ) and the minimum number of points (P_{min}), has a significant impact on the quality of the clustering results. To determine the optimal parameter combination, we start by exploring various values for P_{min} .

Given a specific P_{min} value, we calculate the 75th percentile of the distribution of $(P_{min}/2)$ -nearest distances for the extracted 40% samples with high data Shapley values (as described in Section 4.1 of the paper). We consider this calculated value as the appropriate ϵ for the clustering process. The underlying rationale is that regions with local densities exceeding twice the upper bound of the global density represent distinct high-density areas.

By iterating over different values of P_{min} and adjusting the corresponding ϵ values, we assess the clustering quality achieved by each parameter combination using the Silhouette score, which measures the cohesion and separation of clusters to evaluate their quality. After a thorough evaluation, we determine that a value of P_{min} equal to 100 yields the most suitable parameter choice for our DBSCAN clustering. This method ensures that the clustering process considers the distribution of distances within high-density areas and selects an appropriate value for ϵ , leading to improved clustering results based on the Silhouette score assessment.

E Experimental Set-up

E.1 Hospital-acquired AKI Prediction and Data Processing

Hospital-acquired AKI (short for acute kidney injury) is a disease we strive to handle in our medical practice as front-line clinicians and medical researchers. According to the KDIGO criteria [19], the definition of AKI is based on the rise of sCr (i.e., serum creatinine), a lab test, beyond a threshold limit within a defined timeline. The definition includes two criteria: absolute AKI and relative AKI, as depicted in Figure 2. Absolute AKI is defined as an increase in sCr of more than 26.5 $\mu\text{mol/L}$ within the past two days. Relative AKI, on the other hand, is defined as a rise in sCr of 1.5 times or higher compared to the lowest sCr value within the last seven days.

In hospital-acquired AKI prediction, our goal is to predict whether a patient will develop AKI within two days of hospital admission. We evaluate our approach on our hospital’s EMR data, containing 709 lab tests as input features. Each hospitalized admission in the data is treated as a sample for analysis. In total, we receive 20,732 admissions, with 911 of them resulting in AKI development. We partition the dataset into 90% training data and 10% testing data.

For positive samples where AKI develops during admission, we record the time of AKI detection and define a two-day window, referred to as the “Output Window”, that counts backward from the detection time. This window is not used as input but is crucial in medical practice as it provides a 48-hour lead time, enabling clinicians to take timely interventions following AKI prediction if necessary. The “Input Window”, which serves as input for analysis, spans seven days prior to the Output Window. The relationship between the Input Window and the Output Window is depicted in

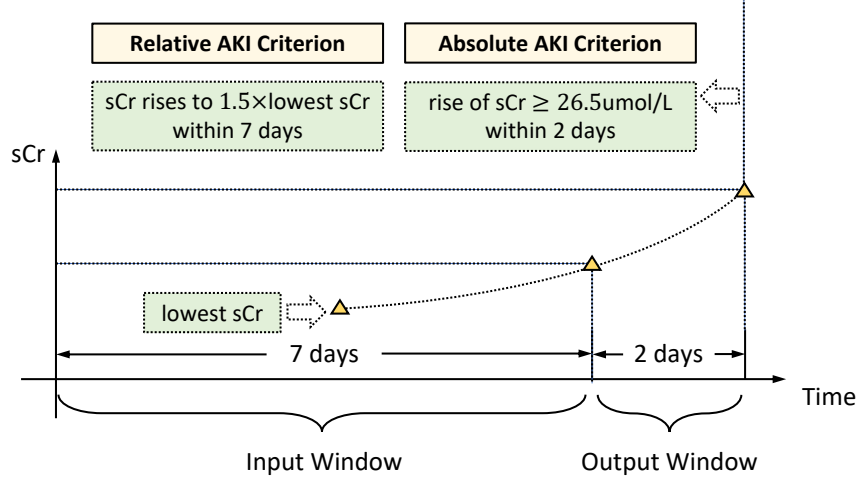


Figure 2: Definition of absolute AKI and relative AKI in hospital-acquired AKI prediction.

Table 2: Key statistics of our dataset for hospital-acquired AKI prediction

Statistics	Our Dataset
# of admissions	20732
# of positive samples	911
# of negative samples	19821
# of lab tests	709
Input Window	7 days
Output Window	2 days

Figure 2. For negative samples, the time of the last recorded lab test is used to determine both the Output Window and the Input Window, respectively.

In summary, our approach utilizes 709 lab tests within the Input Window to predict the likelihood of each sample (i.e., admission) developing AKI after the Output Window. We perform the min-max standardization on the lab test values and then calculate the average to derive input features. Table 2 presents key statistics of our dataset for hospital-acquired AKI prediction.

E.2 Experimental Environment

We conduct the experimental evaluation on a server with the specifications as follows. (i) CPU: Intel(R) Xeon(R) Gold 6248R \times 2, with a clock speed of 3.0GHz and 24 cores per chip. (ii) Memory: the server is equipped with 768GB of memory. (iii) GPU: there are 8 NVIDIA V100 GPUs with NVLINK technology, providing a high-speed data transfer rate of 300GB/s. (iv) Software: The models are implemented using PyTorch version 1.12.1.