Supplementary

Table S1 PCA pseudocode

1. Import necessary libraries.
2. Configure TensorFlow for GPU memory management.
3. Load dataset and set parameters for cross-validation.
4. Load and print data splits.
5. Initialize matrices to store precision values.
6. Set data augmentation parameters.
7. Perform cross-validation:
   a. For each fold:
      i. Load training, validation, and test data.
      ii. Augment the training data.
      iii. Extract training outputs.
      iv. Standardize training, validation, and test data.
      v. Scale the last feature of training and test data.
      vi. Print the shape of the standardized training data.
8. Perform PCA:
   a. Set number of components.
   b. Apply PCA to training and test data.
9. Use Isolation Forest for anomaly detection:
   a. Initialize and fit the model on training data.
   b. Predict anomalies on test data and calculate anomaly scores.
10. Convert predictions to binary (0 for normal, 1 for anomaly).
11. Calculate ROC AUC score using anomaly scores and test labels.
12. Plot PCA-transformed data with anomalies highlighted (optional).
13. Plot ROC curve:
   a. Compute false positive rate, true positive rate, and thresholds.
   b. Calculate ROC AUC.
   c. Determine optimal threshold.
14. Create binary predictions based on optimal threshold.
15. Calculate confusion matrix and evaluation metrics:
   a. Compute confusion matrix.
   b. Calculate true negative rate, F1 score, precision, and recall.
16. Print evaluation metrics.
17. Plot ROC curve.

https://dx.doi.org/10.21037/jmai-24-179

Table S2 GMM pseudocode

1. Import necessary libraries for data manipulation, model building, and evaluation.
2. Configure TensorFlow for GPU memory management.
3. Load dataset and set parameters for cross-validation.
4. Load and print data splits.
5. Initialize matrices to store precision values.
6. Set data augmentation parameters.
7. Perform cross-validation:
   a. For each fold:
      i. Load training, validation, and test data.
      ii. Augment the training data.
      iii. Extract training outputs.
      iv. Standardize training, validation, and test data.
      v. Scale the last feature of training and test data.
      vi. Print the shape of the standardized training data.
8. Perform Gaussian Mixture Model (GMM) fitting:
   a. Set the number of components.
   b. Fit GMM to training data.
   c. Compute anomaly scores on test data using GMM.
9. Calculate ROC AUC score using anomaly scores and test labels.
10. Plot ROC curve:
    a. Compute false positive rate, true positive rate, and thresholds.
    b. Calculate ROC AUC.
    c. Determine the optimal threshold.
11. Create binary predictions based on optimal threshold.
12. Calculate confusion matrix and evaluation metrics:
    a. Compute confusion matrix.
    b. Calculate true negative rate, F1 score, precision, and recall.
13. Print evaluation metrics.

https://dx.doi.org/10.21037/jmai-24-179

Table S3 VAE pseudocode

1. Import necessary libraries for data manipulation, model building, evaluation, and plotting.
2. Load dataset and define parameters for cross-validation.
3. Load data splits using a custom function and print the data.
4. Initialize matrices to store precisions for baseline models and CNN.
5. Set data augmentation parameters based on the dataset.
6. Perform training and validation by cross-validation:
   a. For each fold:
      i. Load training, validation, and test data.
      ii. Augment the training data.
      iii. Extract training outputs into a list.
      iv. Standardize the training, validation, and test data.
      v. Scale the last feature of the training and test data.
      vi. Print the shape of the standardized training data.
7. Define the VAE model:
   a. Set input and latent dimensions.
   b. Create the encoder with input, hidden, mean, and log variance layers.
   c. Create a sampling function.
   d. Create the decoder with hidden and output layers.
   e. Instantiate the VAE model.
8. Define the VAE loss function:
   a. Calculate reconstruction loss.
   b. Calculate KL divergence loss.
   c. Combine losses into the VAE loss.
9. Compile the VAE model with 'adam' optimizer.
10. Train the VAE model on the training data with validation on the test data.
11. Generate new samples from the VAE model.
12. Calculate precision-recall metrics:
   a. Compute reconstruction error.
   b. Calculate precision and recall.
   c. Calculate average precision score.
13. Calculate ROC metrics:
   a. Compute false positive rate, true positive rate, and thresholds.
   b. Calculate ROC AUC.
14. Determine the optimal threshold based on the ROC curve.
15. Create binary predictions based on the optimal threshold.
16. Calculate confusion matrix and evaluation metrics:
   a. Compute confusion matrix.
   b. Calculate true negative rate, F1 score, precision, and recall.
17. Print evaluation metrics: ROC AUC, true positive rate, true negative rate, F1 score, precision, and recall.
18. Plot the precision-recall and ROC curves.

https://dx.doi.org/10.21037/jmai-24-179

Table S4 Anomaly autoencoder pseudocode

1. Import necessary libraries for data manipulation, visualization, and model building.
2. Configure TensorFlow for GPU memory management.
3. Load dataset and set parameters for cross-validation.
4. Load and print data splits.
5. Initialize matrices to store precision values.
6. Set data augmentation parameters.
7. Perform cross-validation:
   a. For each fold:
      i. Load training, validation, and test data.
      ii. Augment the training data.
      iii. Extract training outputs.
      iv. Standardize training, validation, and test data.
      v. Scale the last feature of training and test data.
      vi. Print the shape of the standardized training data.
8. Define an autoencoder model using TensorFlow:
   a. Define the encoder with two dense layers.
   b. Define the decoder with two dense layers.
   c. Combine the encoder and decoder to form the autoencoder.
9. Compile the model with Adam optimizer and mean squared error loss.
10. Train the model on the training data and validate on the test data:
    a. Set batch size and number of epochs.
    b. Plot the training and validation loss over epochs.
11. Use the trained autoencoder to predict on the training data:
    a. Calculate reconstruction error for each training sample.
    b. Store reconstruction errors in a DataFrame along with true class labels.
12. Plot the precision-recall curve:
    a. Calculate precision, recall, and thresholds.
    b. Determine the best threshold to maximize F1 score.
    c. Plot precision and recall for different threshold values.
    d. Plot F1 score for different threshold values.
13. Plot the precision-recall curve with F1 scores:
    a. Annotate F1 score lines.
    b. Highlight the best F1 score point.
14. Plot the ROC curve:
    a. Calculate false positive rate and true positive rate.
    b. Compute ROC AUC.
    c. Plot the ROC curve and display the AUC value.

https://dx.doi.org/10.21037/jmai-24-179