Workflow Document: Semi-Supervised Binary Classification on Software Vulnerability Dataset (BigVul)

# Approach 2: Pseudo-Labeling for Software Vulnerability Detection

## 1. Process Overview

Goal: To classify software vulnerabilities by iteratively training a Graph Convolutional Network (GCN) using both labeled and unlabeled data through pseudo-labeling.  
Algorithm: The approach involves:  
 1. Training an initial GCN model on a small labeled set.  
 2. Applying pseudo-labeling to assign high-confidence labels to unlabeled samples.  
 3. Re-training the model with a larger labeled set, including pseudo-labeled data.

## 2. Data Preparation

1. Relevant Columns: The dataset was filtered to retain only the essential columns related to the function before and after the fix, the lines of code, vulnerability labels, and the function name with fixes.  
 - Columns:  
 - func\_before  
 - func\_after  
 - lines\_before  
 - lines\_after  
 - vul  
 - vul\_func\_with\_fix  
  
2. Handling Missing Data:  
 - The columns lines\_before and lines\_after had missing values, which were filled with empty strings.  
  
3. Class Balance:  
 - The dataset was perfectly balanced with 10,900 samples in each class (vul = 0 and vul = 1).  
  
4. Text Tokenization and Embeddings:  
 - The text data was tokenized using a DistilBERT tokenizer, and the embeddings for each column were generated using the DistilBERT model.  
 - Embeddings were generated for five key columns (func\_before, func\_after, lines\_before, lines\_after, and vul\_func\_with\_fix).  
 - These embeddings were combined into a single representation for each sample using concatenation.

## 3. Preprocessing

1. Dimensionality Reduction:  
 - PCA was applied to reduce the embedding dimensions to 256 components to optimize computation.  
  
2. Normalization:  
 - The embeddings were normalized using L2 normalization to ensure consistency for cosine similarity calculations in FAISS.  
  
3. KNN Graph Construction:  
 - A K-nearest neighbors (KNN) graph was built using FAISS to find the nearest neighbors for each node in the dataset.  
 - K=20 neighbors were selected to define the graph's structure.  
 - The graph was represented as an undirected network and was converted into a PyTorch Geometric data object for further processing.  
  
4. Graph Convolutional Network (GCN):  
 - A simple GCN model was defined with two graph convolution layers, using ReLU activation and dropout for regularization. The model outputs raw logits for binary classification.

## 4. Model Training and Pseudo-Labeling

1. Initial Training:  
 - The model was first trained on the labeled dataset using cross-entropy loss with class weights to handle class imbalance.  
 - Training was done for 50 epochs with early stopping to prevent overfitting.  
   
 Initial Results:  
 Final Validation Accuracy: 0.6720  
 Final Validation Loss: 0.6147  
  
2. Pseudo-Labeling:  
 - After the initial model was trained, pseudo-labeling was applied to assign labels to high-confidence unlabeled nodes, based on a confidence threshold of 0.8.  
 - Nodes with a predicted probability greater than 80% were labeled with their predicted class.  
  
 Results:  
 Pseudo-labeled 3551 nodes.  
  
3. Re-training:  
 - After adding pseudo-labeled data to the training set, the model was retrained for another 50 epochs.  
   
 Retrained Model Performance:  
 Final Validation Accuracy: 0.6720  
 Final Validation Loss: 0.6147

## 5. Performance Evaluation

1. Test Set Evaluation:  
 - After training, the model was evaluated on the original test set (previously unlabeled).  
  
 Test Accuracy: 0.6804  
  
 Confusion Matrix:  
 [[6259 3551]  
 [2719 7091]]  
  
 Classification Report:  
 precision recall f1-score support  
 0 0.6971 0.6380 0.6663 9810  
 1 0.6663 0.7228 0.6934 9810  
 accuracy 0.6804 19620  
 macro avg 0.6817 0.6804 0.6799 19620  
 weighted avg 0.6817 0.6804 0.6799 19620  
  
2. ROC-AUC Curve:  
 - The ROC-AUC was calculated for class 1, indicating the model’s ability to distinguish between vulnerable and non-vulnerable samples.  
 ROC-AUC Score: 0.68

## 6. Insights and Discussion

1. Improvement with Pseudo-Labeling:  
 - The initial model achieved an accuracy of 0.6720. After applying pseudo-labeling, additional confident samples were added to the training set, resulting in an improvement in test accuracy to 0.6804.  
  
2. Challenges:  
 - Class 0 (non-vulnerable) had a lower recall (0.6380) compared to Class 1 (0.7228). This indicates that the model struggled to correctly classify all non-vulnerable samples, leading to more false negatives in this class.  
  
3. Threshold Sensitivity:  
 - The threshold of 0.8 for pseudo-labeling was used to ensure only high-confidence predictions were included. Tuning this threshold could further improve the model's performance by balancing the number of added pseudo-labels with their accuracy.