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

# Approach 3: Label Propagation for Software Vulnerability Detection

## 1. Process Overview

**Goal**: To classify software vulnerabilities using the Label Propagation algorithm in a semi-supervised learning setup. This approach leverages both labeled and unlabeled data by constructing a KNN graph based on feature embeddings and propagating labels across this graph structure.

**Problem with Oversampled Data**: Initial attempts with oversampled data produced embeddings that were not well-separated between classes, as shown in the t-SNE visualization. This lack of separation negatively impacted the performance of the label propagation model.

**Updated Approach with Undersampled Data**: After switching to an undersampled dataset, the model performance significantly improved, achieving high accuracy and well-separated class clusters, as evident in the updated t-SNE visualization.

## 2. Data Preparation

**Dataset**:

* The dataset was balanced by undersampling, resulting in a total of 21,800 records with equal representation of both classes.
* **Columns**: Relevant columns include func\_before, func\_after, lines\_before, lines\_after, vul, and vul\_func\_with\_fix.

**Handling Missing Values**:

* Missing values in textual columns were replaced with empty strings to ensure compatibility during embedding generation.

**Embedding Generation**:

* The textual data was tokenized and converted into embeddings using the DistilBERT model. This process was parallelized across TPU cores to optimize processing time.

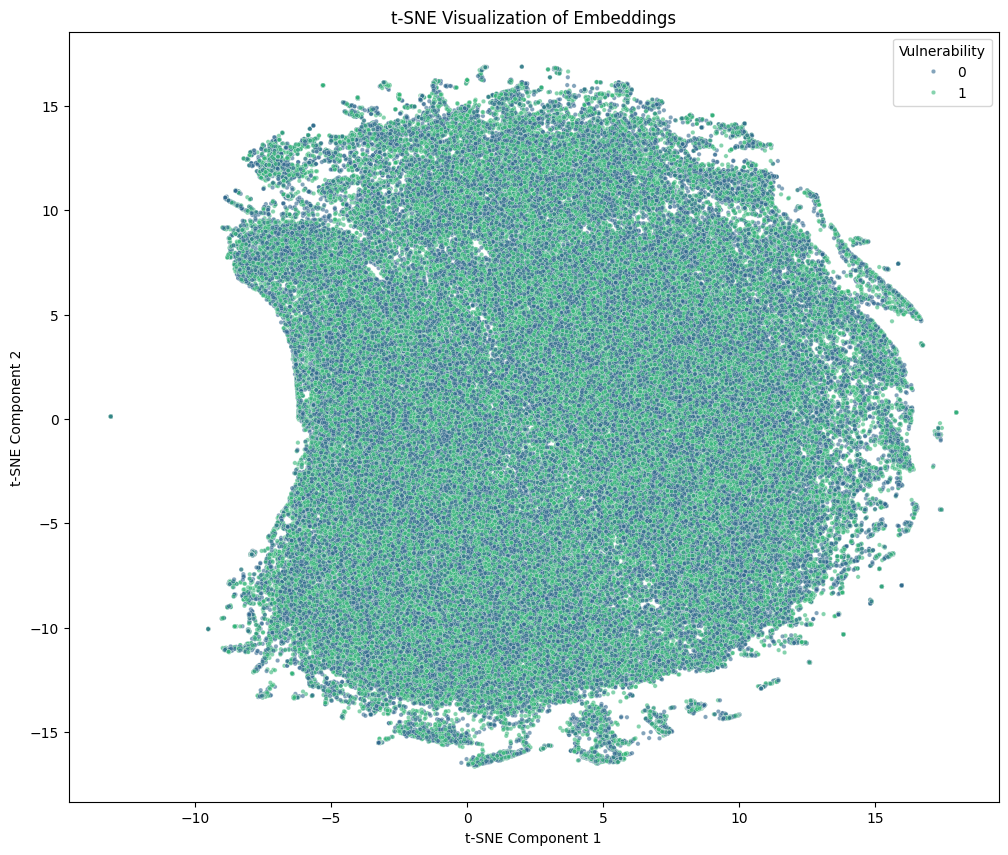
**Dimensionality Reduction and Normalization**:

* PCA was applied to reduce the embeddings to 256 dimensions, capturing 95.95% of the variance. The reduced embeddings were further normalized using L2 normalization.

## 3. Preprocessing and Dimensionality Reduction

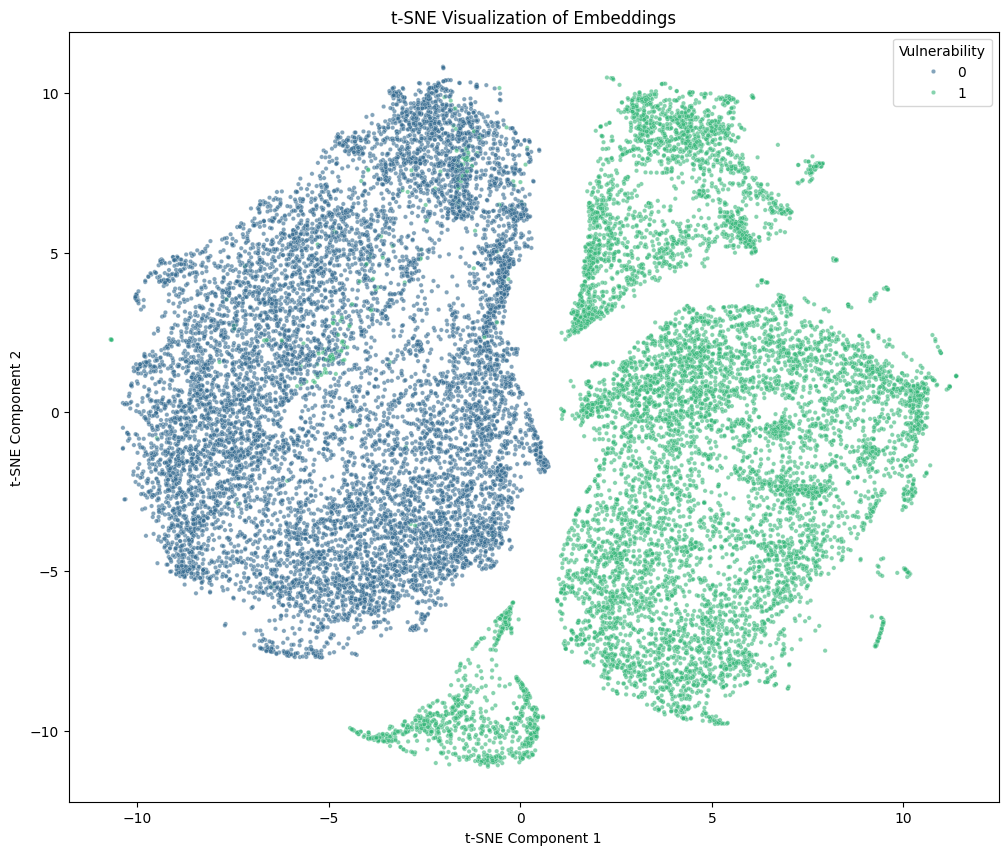
1. Dimensionality Reduction:  
 - PCA (Principal Component Analysis) was applied to reduce the high-dimensional embeddings (from DistilBERT) down to 256 components. This helps to manage computational complexity while retaining important features.  
   
2. Normalization:  
 - The reduced embeddings were normalized using L2 normalization to ensure that they have unit length, which is necessary for cosine similarity calculations during the KNN graph construction.

## t-SNE Visualization on Oversampled Data

Below is the t-SNE visualization for the oversampled embeddings, showing poor class separation: 

## Updated t-SNE Visualization for Undersampled Data

After switching to the undersampled dataset, the t-SNE visualization shows well-separated clusters for the two classes, indicating improved feature representation for label propagation.



## 4. Graph Construction

**KNN Graph Creation:**

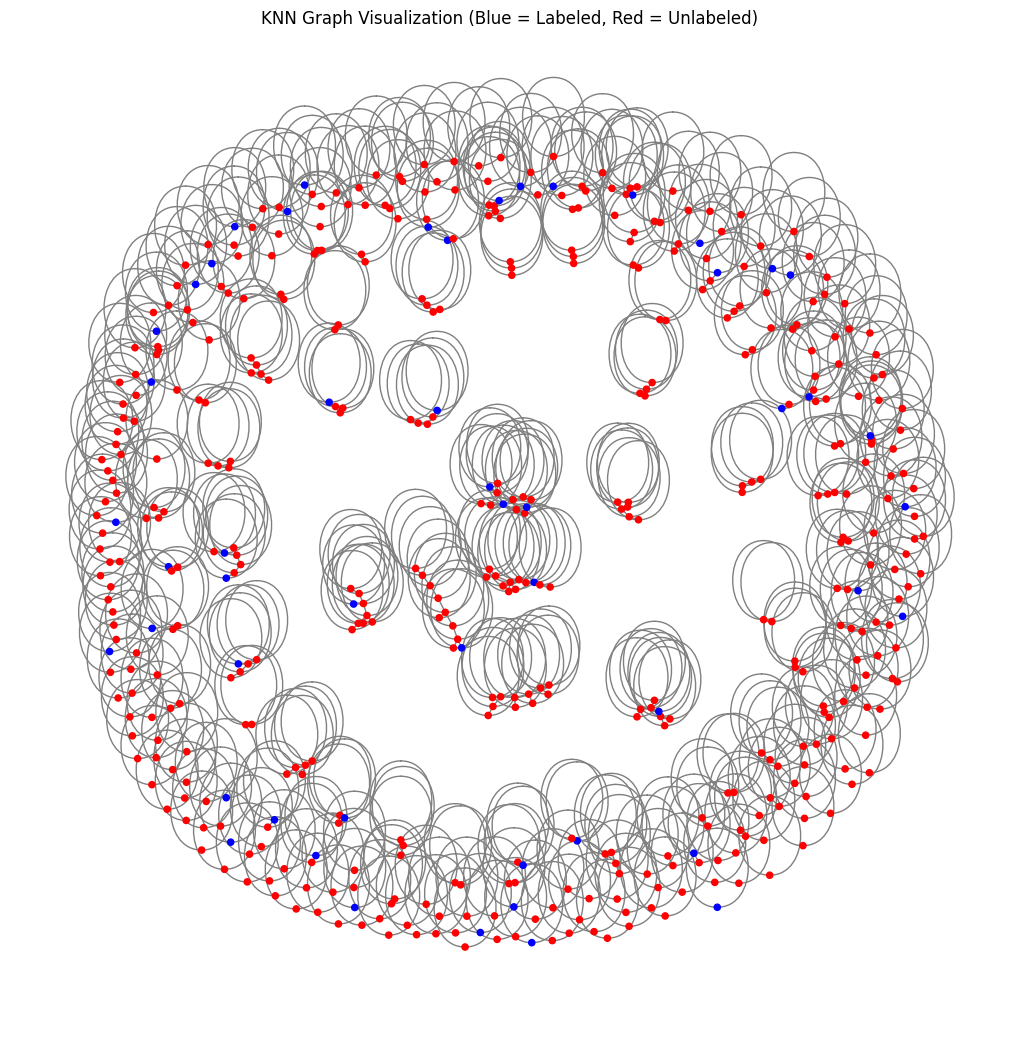
A K-Nearest Neighbors (KNN) graph was constructed with K=10, using FAISS for efficient nearest-neighbor search. Each node connects to its 10 nearest neighbors, and self-loops were added to the graph for each node.

**Gaussian Noise:**

Gaussian noise was added to the embeddings to enhance the model’s robustness.

**KNN Graph Visualization:**

The KNN graph shows labeled nodes (blue) and unlabeled nodes (red), with connections representing neighborhood relations.

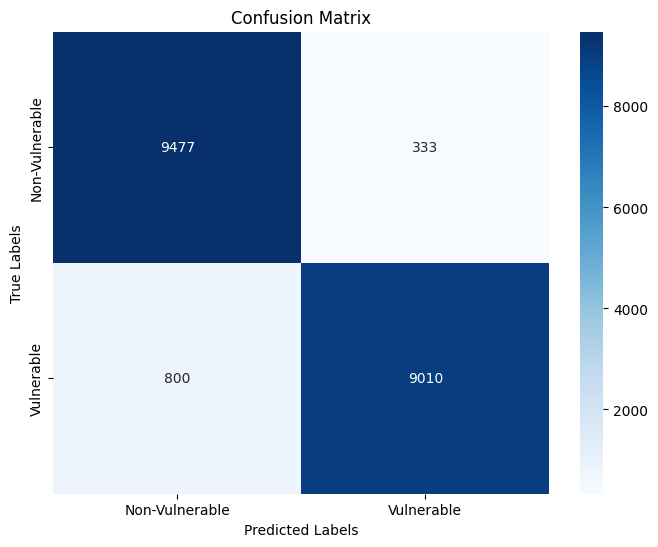


## 5. Label Propagation Model Training

1. Semi-supervised Setup:
   * 10% of the nodes were labeled, and the rest were unlabeled to simulate a semi-supervised environment. This setup allows labels to propagate from the labeled to unlabeled nodes based on neighborhood connections.
2. Label Propagation:
   * The Label Propagation model from PyTorch Geometric was used, with a smoothing parameter alpha=0.6 and 20 iterations. During propagation, labeled nodes retain their original labels while labels are propagated to unlabeled nodes.

## 6. Model Evaluation

1. **Test Set Accuracy**:
   * The model achieved a test accuracy of **94.23%** on the unlabeled set.
2. **Confusion Matrix**:
   * The confusion matrix shows the model’s performance across classes, indicating a strong ability to correctly classify both vulnerable and non-vulnerable samples.



1. **Classification Report**:

The classification report shows precision, recall, and F1-scores for each class, reflecting the model's balanced performance across classes.

Classification Report:

precision recall f1-score support

Non-Vulnerable 0.922156 0.966055 0.943595 9810.000000

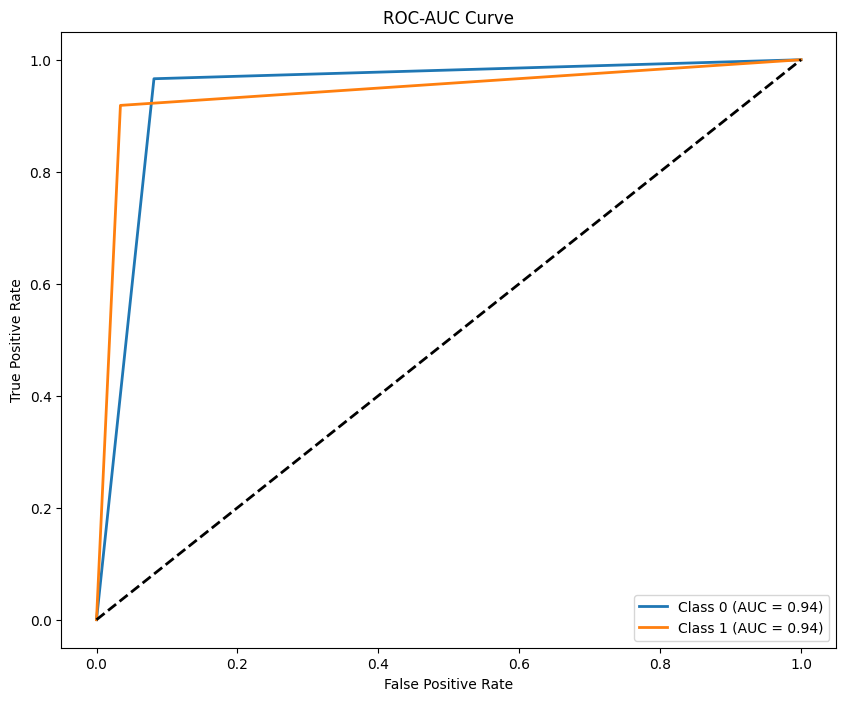
Vulnerable 0.964358 0.918451 0.940845 9810.000000

accuracy 0.942253 0.942253 0.942253 0.942253

macro avg 0.943257 0.942253 0.942220 19620.000000

weighted avg 0.943257 0.942253 0.942220 19620.000000

1. **ROC-AUC Curve**:
   * The ROC-AUC curves for each class (0 and 1) demonstrate the model's ability to distinguish between vulnerable and non-vulnerable samples, with AUC scores of 0.94 for both classes.



## 7. Validation and Consistency Checks

1. **Feature Variance**:
   * Min Variance: 0.0393
   * Max Variance: 0.4596
   * Mean Variance: 0.0439
2. **Nearest Neighbor Distance Distribution**:
   * Min Distance: 13.7616
   * Max Distance: 20.9275
   * Mean Distance: 16.5698
3. **Label Consistency in Neighborhoods**:
   * The average label consistency in local neighborhoods is 0.8867, indicating a high level of similarity between nodes and their neighbors.

These checks confirm the model's robustness and the integrity of the graph structure.