भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Indian Institute of Technology Hyderabad

Fraud Analytics (CS6890)

Assignment - 3

Identifying outliers in the data by using Variational Autoencoders

Name	Roll Number
Saumay Lunawat	CS24MTECH14005
Shivendra Deshpande	CS24MTECH12017
Chinmay Rajesh Ingle	CS23MTECH12002

Contents

List of Figures

1 Problem Statement

The primary objective is to identify outliers or anomalous data points within a given dataset. The specified methodology involves a two-stage approach:

1. **Dimensionality Reduction:** Utilize a Variational Autoencoder (`VAE`) to compress the high-dimensional input data into a lower-dimensional latent space representation. This aims to capture the most salient features and underlying structure of the data while reducing noise and complexity.
2. **Clustering and Outlier Identification:** Apply the `K-Means` clustering algorithm to the generated low-dimensional latent space data. Outliers are then identified based on their position relative to the formed clusters. Specifically, points that lie significantly far from the center (centroid) of their assigned cluster are flagged as outliers. This corresponds to identifying boundary points or points on the periphery of the main data distribution within the latent space. The provided implementation (`assignment_3.py`) uses a distance percentile threshold (e.g., 95th percentile) to define "significantly far".

(Note: The original problem description mentioned two criteria for outliers: "Boundary points of big clusters" and "Points in small clusters". The provided code primarily implements the first criterion by identifying points with large distances to their cluster center, irrespective of cluster size. It does not explicitly isolate points belonging to small clusters as a separate outlier category.)

2 Dataset Description

2.1 Source Data (Implicit)

The analysis pipeline starts with an input dataset presumed to be named `data.csv` (as used in the `if` *__name__* == "*__main__*" : block of the script).

Features: Based on the header of the `outliers.csv` file and the code's selection of numeric types, the original dataset contains at least 10 numerical features:

- `cov1, cov2, cov3, cov4, cov5, cov6, cov7` (likely representing covariance or covariate features)
- `sal_pur_rat` (likely sales-purchase ratio)
- `igst_itc_tot_itc_rat` (likely related to GST Input Tax Credit ratios)
- `lib_igst_itc_rat` (likely related to liability or library and GST Input Tax Credit ratios)

2.2 Preprocessing:

The `load_data` function in the script reads the data, selects only numeric columns, and then applies `StandardScaler`. This normalization step standardizes features by removing the mean and scaling to unit variance, crucial for algorithms like VAE and K-Means.

2.3 Output Data (`outliers.csv`):

The provided `outliers.csv` file represents the *result* of the analysis. It contains the subset of data points identified as outliers (58 points in the provided file).

- It includes the original 10 feature values for each outlier.
- It adds three columns generated by the analysis: An unnamed index column (original row index), `distance_to_center` (Euclidean distance in latent space to cluster centroid), and `cluster` (assigned K-Means label: 0, 1, or 2).

3 Algorithm

The solution employs three core algorithmic components:

1. **Variational Autoencoder (VAE):**
    - *Purpose:* Unsupervised deep learning for dimensionality reduction and generative modeling.
    - *Architecture:* Consists of two neural networks:
        - **Encoder:** Maps input `x` to latent distribution parameters (`z_mean`, `z_log_var`) and samples a latent vector `z`.
        - **Decoder:** Maps latent vector `z` back to reconstruct input `x`.
    - *Training:* Minimizes Reconstruction Loss + **KL Divergence Loss** (regularizer).
    - *Usage:* The trained encoder transforms preprocessed data into lower-dimensional `latent_data` (specifically 2D).
2. **K-Means Clustering:**
    - *Purpose:* Unsupervised partitioning of observations into `k` clusters.
    - *Mechanism:* Iteratively assigns points to the nearest cluster centroid (Euclidean distance) and recalculates centroids until convergence.
    - *Usage:* Applied to the VAE's `latent_data`. The optimal number of clusters (`k`) is determined using the **ELBOW METHOD** (distortion) and silhouette score evaluation, leading to $k = 3$.
3. **Outlier Detection Logic:**
    - *Method:* Distance-based outlier detection in the latent space.
    - *Mechanism:* Calculate Euclidean distance $d(p, center_c)$ for each latent point p to its assigned cluster c's centroid $center_c$. A threshold is set at the 95th percentile (`percentile_threshold=95`) of these distances. Points exceeding this threshold are flagged as outliers.
    - *Rationale:* Points far from their cluster center in the latent space are less representative of typical patterns.

4 Approach

The `assignment_3.py` script implements the following pipeline (`run_vae_clustering_pipeline` function):

1. **Load and Preprocess Data:**
    - Read from `data.csv` using `load_data`.
    - Select numeric features.
    - Normalize using `StandardScaler` to get `normalized_data`.
2. **VAE Training and Evaluation (Experimentation):**
    - Restricted to run only for `latent_dims=[2]`.
    - Train `VAE` model (`train_vae`) with `epochs=50`, `batch_size=32`.
    - Generate 2D `latent_data` using the encoder.
    - Determine optimal cluster count `k` using `find_optimal_clusters` (distortion) and `evaluate_clusters` (K-Means + `silhouette_score`).

- Select the best `k` (found to be 3) based on the highest silhouette score for `latent_dim=2`. Store the best K-Means model (`best_kmeans`), score, latent data (`best_latent_data`), and VAE model (`best_vae`).

3. **Outlier Detection:**
   - Use `best_latent_data` and `best_kmeans`.
   - Calculate distances to centroids using `find_outliers`.
   - Determine threshold using `percentile_threshold=95`.
   - Identify outlier indices (`boundary_points`).
   - Create `outlier_df` with original features, distances, cluster labels.
   - Save to `outliers.csv`.
4. **Results Reporting and Visualization:**
   - Print summary: best config, outlier count/percentage.
   - Display top 5 outliers (`display_outlier_info`).
   - Generate 2D scatter plot (`visualize_clusters_and_outliers`) of `best_latent_data`:
     - Color points by cluster (`0, 1, 2`).
     - Mark outliers (red circles).
     - Show centroids (black 'X').
   - Save plot as `clusters_2d.png`.

5 Results & Observations

- **Optimal Configuration:** The pipeline identified the best configuration as:
  - Latent Dimension: 2
  - Number of Clusters: 3
- **Outlier Identification:** 58 data points were identified as outliers using the 95th percentile distance threshold in the 2D latent space. These are detailed in `outliers.csv`.
- **Latent Space Visualization (`clusters_2d.png`):**
  - The plot (Figure **??**) displays the data projected onto the 2D latent space.
  - Three clusters (`Cluster 0`, `Cluster 1`, `Cluster 2`) are shown, with some overlap observed.
  - Identified outliers (red circles) are primarily located on the periphery of their clusters, consistent with the distance-based detection.
  - Cluster centers (black 'X') mark the average position within each cluster in the latent space.
- **`outliers.csv` Data:** This file provides the original feature values for the 58 outliers, allowing for investigation into *why* they were flagged. The `distance_to_center` column quantifies their deviation in the latent space.
- **Interpretation:** The VAE successfully reduced dimensionality, and K-Means grouped the data in the latent space. The identified outliers represent points statistically distant from the core patterns learned by the model, potentially indicating anomalies, errors, or rare events that warrant further examination using their original feature values from `outliers.csv`.

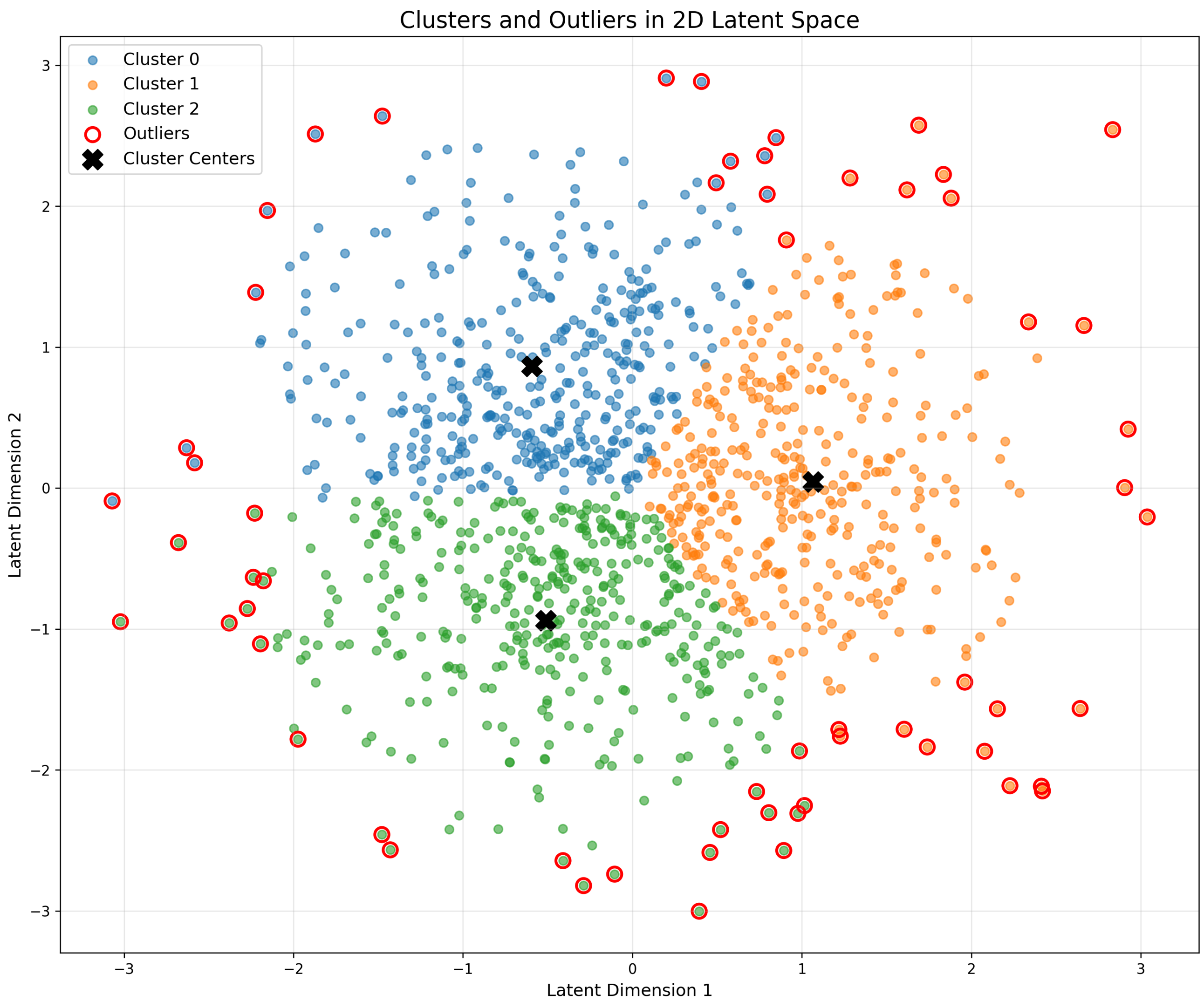

Figure 1: Clusters