Fraud Analytics - Assignment 3

Outlier Detection using Variational Autoencoders (VAE) and K-Means

Sakshi Badole CS24MTECH11008

Laveena Herman CS24MTECH11010

S Anjana Shankar CS24MTECH14015

April 6, 2025

1 Introduction

Identifying outliers is an essential aspect of data analysis, as these anomalous data points may signify unusual occurrences, data corruption, or new patterns in data. Outliers are data points that deviate significantly from the predominant distribution. Variational Autoencoders (VAEs) are advanced deep learning models that excel in learning latent representations, which captures the underlying structure of the data. In this study, we explore the use of Variational Autoencoder (VAE) for dimensionality reduction, followed by k-means clustering method to detect outliers present in the dataset.

2 Variational Autoencoders (VAE)

2.1 VAE Architecture

A Variational Autoencoder (VAE) is a generative model that learns to encode input data into a probabilistic latent space and then reconstructs it back. Unlike traditional autoencoders, which directly learn a deterministic mapping, VAEs learn a probability distribution over the latent space, making them powerful for data generation.
A Variational Autoencoder (VAE) consists of three main components:

1. Encoder (Inference Network)

The encoder maps the input x to a latent distribution instead of a fixed point. It outputs a mean μ and variance σ^2 of a latent Gaussian distribution:

$$q(z|x) = \mathcal{N}(\mu, \sigma^2) \tag{1}$$

The encoder approximates the true posterior distribution.

2. Latent Space (Stochastic Sampling)

Instead of a deterministic latent representation, we sample from the learned distribution using the **Reparameterization Trick**:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1) \tag{2}$$

This allows gradients to propagate during training.

3. Decoder (Generative Network)

The decoder reconstructs x' from the latent variable z. It learns the likelihood distribution:

$$p(x|z) \tag{3}$$

to generate data similar to x.

2.2 VAE Loss Function

A Variational Autoencoder (VAE) optimizes a combination of two losses:

Reconstruction Loss

The Reconstruction Loss ensures that the reconstructed output x' is close to the original input x. It is commonly computed using Mean Squared Error (MSE):

$$\mathcal{L}_{\text{recon}} = \mathbb{E}_{q(z|x)} \left[\|x - x'\|^2 \right] \tag{4}$$

KL Divergence Loss

The KL Divergence Loss forces the learned latent distribution $q(z|x)$ to be close to a standard normal distribution $p(z)$, which encourages better generalization and prevents overfitting:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}} \left(q(z|x) \| p(z) \right) \tag{5}$$

For Gaussian distributions, the closed-form solution for KL divergence is:

$$\mathcal{L}_{\text{KL}} = -\frac{1}{2} \sum \left(1 + \log \sigma^2 - \mu^2 - \sigma^2 \right) \tag{6}$$

Total VAE Loss

The total loss function for VAE is given by:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \beta \mathcal{L}_{\text{KL}} \tag{7}$$

where the term β in β-VAE controls the weight of the KL loss, which can help in better disentanglement of latent features.

2.3 VAE for dimension reduction

Variational Autoencoders (VAEs) reduce high-dimensional data to a lower-dimensional latent space while preserving meaningful structure. The encoder maps inputs to a probabilistic distribution, from which a latent representation is sampled using the reparameterization trick. The decoder then reconstructs the original data from this latent space. The training process optimizes a balance between reconstruction loss (to retain data fidelity) and KL divergence (to ensure a structured latent space), allowing for effective and meaningful compression.

Traditional PCA relies on linear projections, limiting its ability to capture complex data relationships. In contrast, VAEs learn a probabilistic, non-linear mapping from input x to a latent variable z, enforcing a structured representation, enabling better data reconstruction through deep networks while enforcing a smooth, structured representation via KL divergence. This results in more interpretable features that correspond to meaningful variations in the data, such as shape, style, or pose in images.

2.4 K-means clustering

K-Means is an unsupervised machine learning algorithm used for clustering, where data points are grouped into k clusters based on similarity. It minimizes intra-cluster variance by iteratively refining cluster assignments. It first initializes k cluster centroids and then assigns each data point to the nearest centroid based on the Euclidean distance between the point and the centroids. The centroids are then updated as the mean of the assigned points, and the process repeats until convergence. The algorithm minimizes intra-cluster variance, making it effective for grouping similar data points. To determine the optimal number of clusters for grouping, Elbow Method can be used.

3 Algorithm for Outlier Detection Using VAE and K-Means

3.1 Data Preprocessing

- Normalize the dataset using Standard Scaler to ensure all features have zero mean and unit variance.

3.2 Dimensionality Reduction with VAE

- Train a Variational Autoencoder (VAE) on the dataset to learn a lower-dimensional latent space.
- Encode the input data into the latent space by extracting the mean representation μ.

3.3 Clustering with K-Means

- Apply the K-Means clustering algorithm on the latent representations.
- Determine the optimal number of clusters k using the Elbow Method.

3.4 Dimension Selection and Evaluation

- Repeat the process for multiple latent space dimensions and different values of k.
- Evaluate models using:
  - Silhouette Score and DB Index to measure cluster separation.
  - Reconstruction Loss generated by VAE to assess how well VAE retains information.
  - t-SNE visualization to analyze latent space structure.
- Select the best latent dimensions which produces a better representation and number of clusters.

3.5 Outlier Identification

- Identify the outliers in the following 2 ways:
  - Identify boundary points in large clusters by calculating the distance of each point from the cluster centroid; points farther from the centroid are potential outliers.
  - Detect small clusters, as they may represent rare or anomalous patterns in the data.
- Mark data points belonging to these 2 categories as outliers.

3.6 Common outliers

- Using the best dimensions and number of clusters, outliers are detected and the common outliers among them is selected as the final list of outliers.

3.7 Outlier Verification

- The outliers identified to be verified through common outlier detection algorithms such as z-score and isolation forest.

4 Results and Discussion

4.1 Dimension selection and evaluation

Various latent dimensions and number of clusters were tried on and analyzed on different methods to select the best combination.

4.1.1 Silhouette Score and Davies Bouldin Index

For each latent dimension, the k value which gives the highest silhouette score and db index was found out and the best silhouette and db score were noted and plotted against the latent dimension.

Latent Dimension	Silhouette Score	DB Score
2	0.491	0.953
3	0.264	1.458
4	0.951	0.554
5	0.279	1.701
6	0.141	2.245
7	0.164	1.948

Table 1: Clustering Quality Metrics (Silhouette and DB Scores) for Different Latent Dimensions

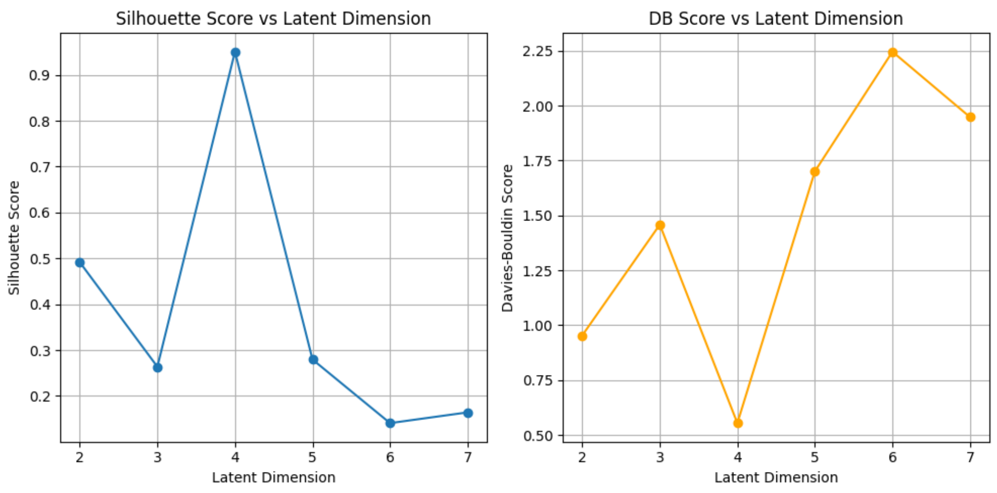


Figure 1: Silhouette Score and Davies-Bouldin Index vs Latent Dimension

The higher the silhouette score and the lower the db index, the better. Hence, from the plots it can be seen that dimension 4 shows better values.

4.1.2 Elbow Method

Using the elbow method, the best value of number of clusters was plotted and found out for different dimensions.

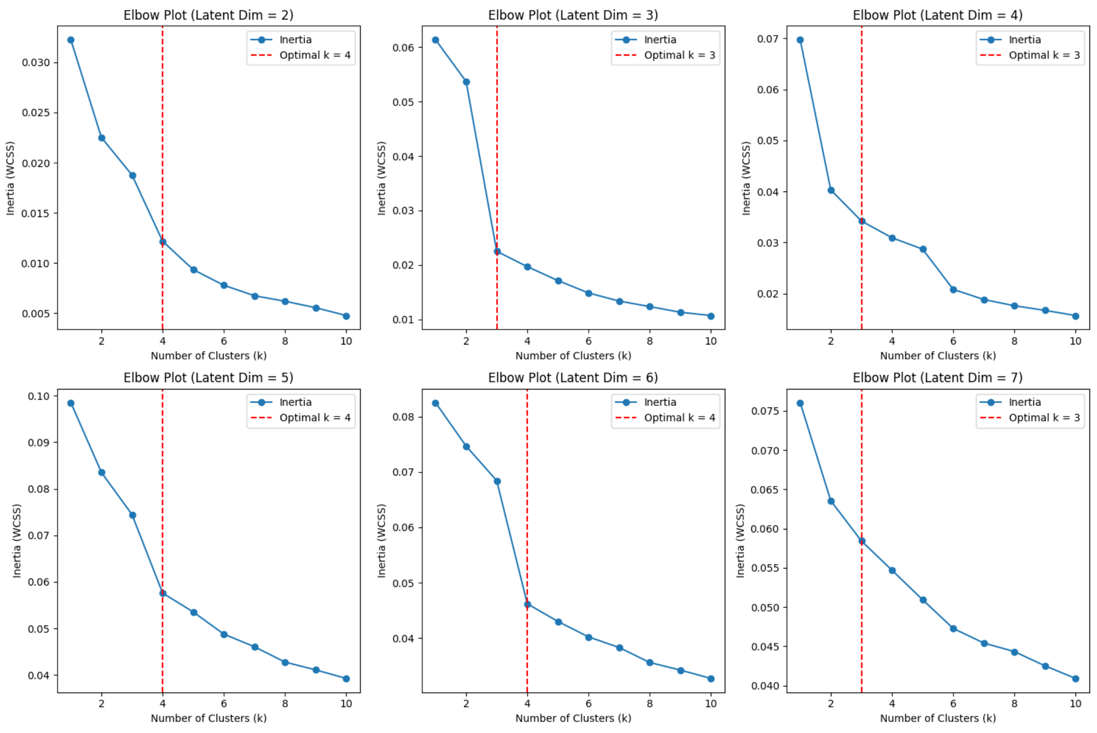


Figure 2: Elbow Method

4.1.3 Reconstruction Loss of VAE

The reconstruction loss in each dimension was found out and was plotted against each dimension to find out which shows the least loss.

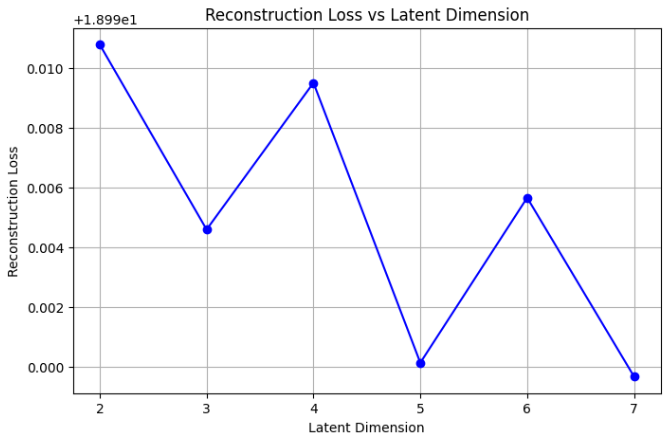


Figure 3: Reconstruction Loss vs Latent Dimension

From the plots, dimensions 3,5,7 showed lesser reconstruction errors. Dimension 2 has high reconstruction error.

4.1.4 t-sne plot

The t-sne plot was plotted for different dimensions to analyze the latent space structure.

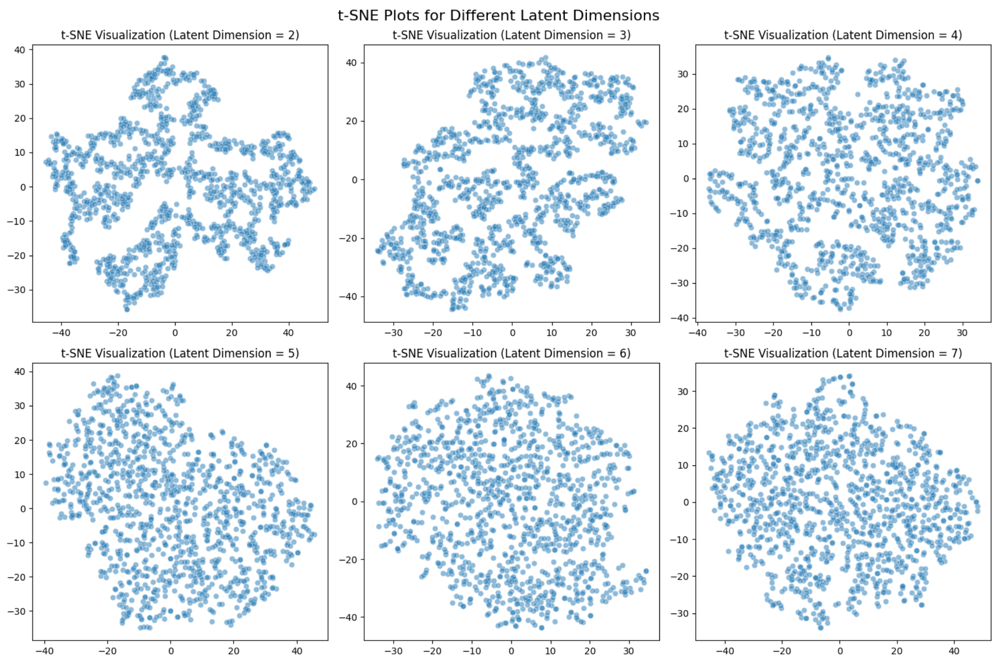


Figure 4: t-sne plot for different Latent Dimension

From the plots, it can be seen that dimensions 2, 3 and 4 shows better representations.

4.1.5 Inference

Based on the above analysis, dimensions 3, 4, and 5 produced the best latent representations. The corresponding best values for the number of clusters were found out using the Elbow method. The values are:

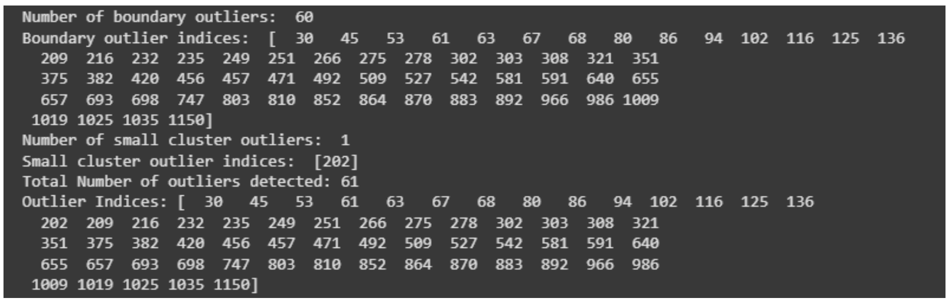
Latent Dimension	**Optimal Number of Clusters**
3	3
4	3
5	4

4.2 Model training and Outliers detection

- The latent dimension reduction for dimensions 3, 4 and 5 was done and kmeans was applied on the latent representations.
- The beta for VAE loss calculation was set to 1.5 to give more weightage to KL divergence loss to produce a better generalized structure for the latent dimension representations.
- The threshold percentage radius for finding out the boundary outliers was set to 95
- The model was trained for 50 epochs.
- The outliers were detected (both boundary as well as small cluster outliers) from the clusters formed after applying kmeans on the latent representations. The results are as follows:

4.2.1 Dimension 3

Below are the results for dimension 3, the best k was found to be at 3. The following are the outlier indices detected:

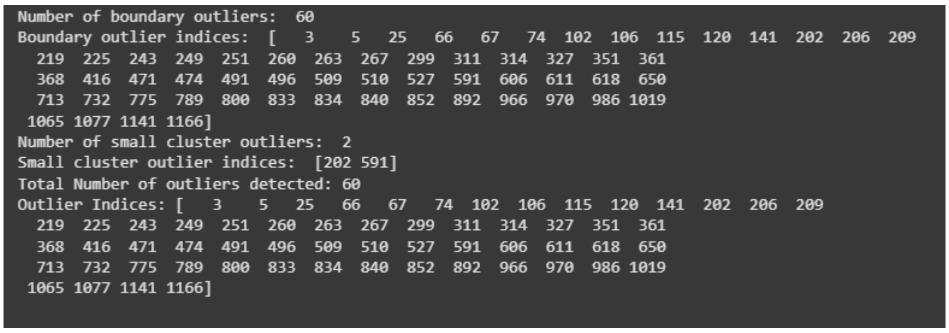


```
Number of boundary outliers:  60
Boundary outlier indices:  [  30   45   53   61   63   67   68   80   86   94  102  116  125  136
  209  216  232  235  249  251  266  275  278  302  303  308  321  351
  375  382  420  456  457  471  492  509  527  542  581  591  640  655
  657  693  698  747  803  810  852  864  870  883  892  966  986 1009
 1019 1025 1035 1150]
Number of small cluster outliers:  1
Small cluster outlier indices:  [202]
Total Number of outliers detected: 61
Outlier Indices: [  30   45   53   61   63   67   68   80   86   94  102  116  125  136
  202  209  216  232  235  249  251  266  275  278  302  303  308  321
  351  375  382  420  456  457  471  492  509  527  542  581  591  640
  655  657  693  698  747  803  810  852  864  870  883  892  966  986
 1009 1019 1025 1035 1150]
```

Figure 5: Outlier indices for latent dimension 3

4.2.2 Dimension 4

Below are the results for dimension 4, the best k was found to be at 3. The following are the outlier indices detected:

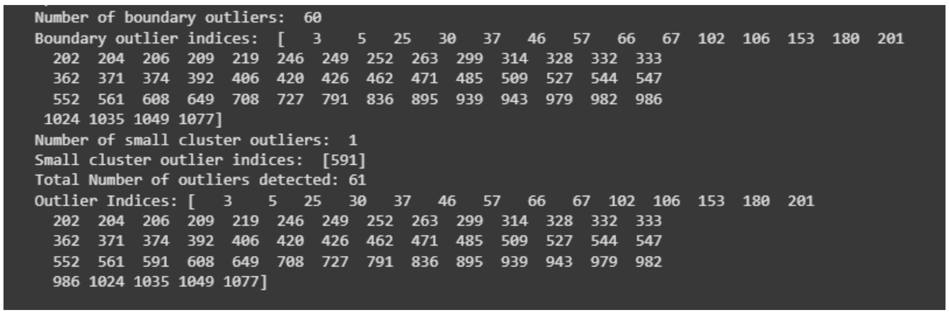


```
Number of boundary outliers:  60
Boundary outlier indices:  [   3    5   25   66   67   74  102  106  115  120  141  202  206  209
  219  225  243  249  251  260  263  267  299  311  314  327  351  361
  368  416  471  474  491  496  509  510  527  591  606  611  618  650
  713  732  775  789  800  833  834  840  852  892  966  970  986 1019
 1065 1077 1141 1166]
Number of small cluster outliers:  2
Small cluster outlier indices:  [202 591]
Total Number of outliers detected: 60
Outlier Indices: [   3    5   25   66   67   74  102  106  115  120  141  202  206  209
  219  225  243  249  251  260  263  267  299  311  314  327  351  361
  368  416  471  474  491  496  509  510  527  591  606  611  618  650
  713  732  775  789  800  833  834  840  852  892  966  970  986 1019
 1065 1077 1141 1166]
```

Figure 6: Outlier indices for latent dimension 4

4.2.3 Dimension 5

Below are the results for dimension 5, the best k was found to be at 4. The following are the outlier indices detected:

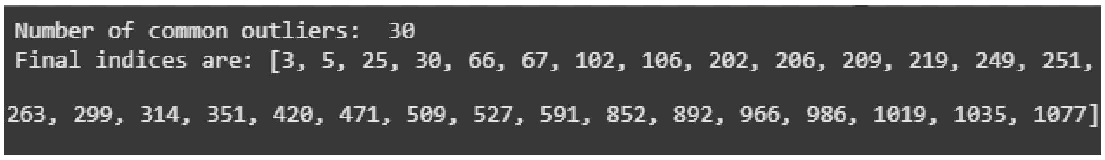


```
Number of boundary outliers:  60
Boundary outlier indices:  [   3    5   25   30   37   46   57   66   67  102  106  153  180  201
  202  204  206  209  219  246  249  252  263  299  314  328  332  333
  362  371  374  392  406  420  426  462  471  485  509  527  544  547
  552  561  608  649  708  727  791  836  895  939  943  979  982  986
 1024 1035 1049 1077]
Number of small cluster outliers:  1
Small cluster outlier indices:  [591]
Total Number of outliers detected: 61
Outlier Indices: [   3    5   25   30   37   46   57   66   67  102  106  153  180  201
  202  204  206  209  219  246  249  252  263  299  314  328  332  333
  362  371  374  392  406  420  426  462  471  485  509  527  544  547
  552  561  591  608  649  708  727  791  836  895  939  943  979  982
  986 1024 1035 1049 1077]
```

Figure 7: Outlier indices for latent dimension 5

4.2.4 Final outlier indices

The outlier indices which are repeating more than once were selected from the outputs of the 3 different dimensions and reported as the final outliers. The final outlier indices are:

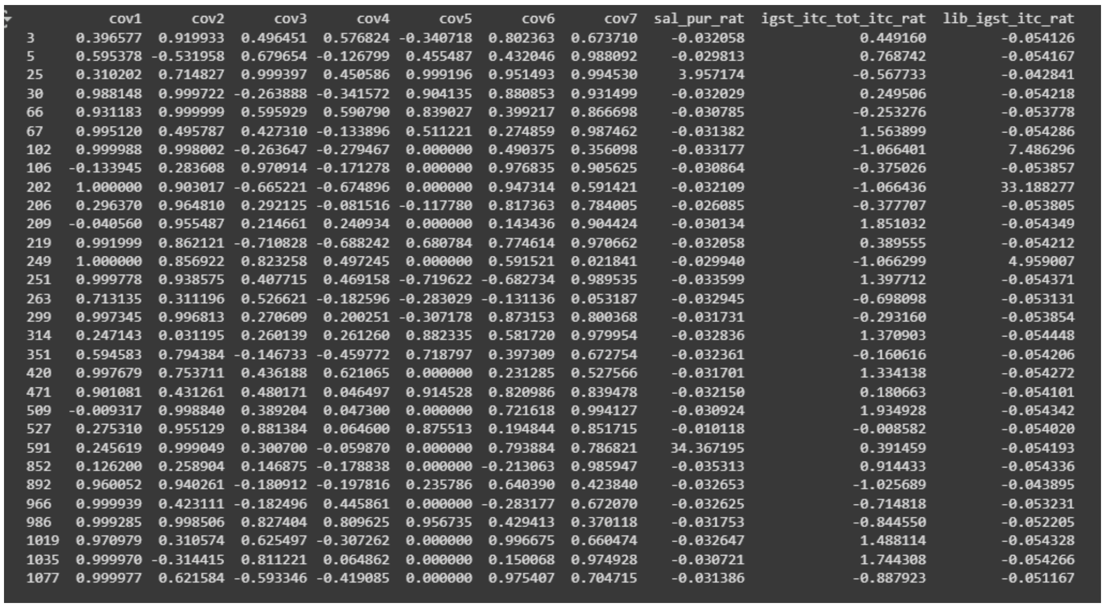


```
Number of common outliers:  30
Final indices are: [3, 5, 25, 30, 66, 67, 102, 106, 202, 206, 209, 219, 249, 251,

263, 299, 314, 351, 420, 471, 509, 527, 591, 852, 892, 966, 986, 1019, 1035, 1077]
```

Figure 8: Outlier indices

The following table shows the outlier values:



```
          cov1      cov2      cov3      cov4      cov5      cov6      cov7  sal_pur_rat  igst_itc_tot_itc_rat  lib_igst_itc_rat
3     0.396577  0.919933  0.496451  0.576824 -0.340718  0.802363  0.673710    -0.032058              0.449160         -0.054126
5     0.595378 -0.531958  0.679654 -0.126799  0.455487  0.432046  0.988092    -0.029813              0.768742         -0.054167
25    0.310202  0.714827  0.999397  0.450586  0.999196  0.951493  0.994530     3.957174             -0.567733         -0.042841
30    0.988148  0.999722 -0.263888 -0.341572  0.904135  0.880853  0.931499    -0.032029              0.249506         -0.054218
66    0.931183  0.999999  0.595929  0.590790  0.839027  0.399217  0.866698    -0.030785             -0.253276         -0.053778
67    0.995120  0.495787  0.427310 -0.133896  0.511221  0.274859  0.987462    -0.031382              1.563899         -0.054286
102   0.999988  0.998002 -0.263647 -0.279467  0.000000  0.490375  0.356098    -0.033177             -1.066401          7.486296
106  -0.133945  0.283608  0.970914 -0.171278  0.000000  0.976835  0.905625    -0.030864             -0.375026         -0.053857
202   1.000000  0.903017 -0.665221 -0.674896  0.000000  0.947314  0.591421    -0.032109             -1.066436         33.188277
206   0.296370  0.964810  0.292125 -0.081516 -0.117780  0.817363  0.784005    -0.026085             -0.377707         -0.053805
209  -0.040560  0.955487  0.214661  0.240934  0.000000  0.143436  0.904424    -0.030134              1.851032         -0.054349
219   0.991999  0.862121 -0.710828 -0.688242  0.680784  0.774614  0.970662    -0.032058              0.389555         -0.054212
249   1.000000  0.856922  0.823258  0.497245  0.000000  0.591521  0.021841    -0.029940             -1.066299          4.959007
251   0.999778  0.938575  0.407715  0.469158 -0.719622 -0.682734  0.989535    -0.033599              1.397712         -0.054371
263   0.713135  0.311196  0.526621 -0.182596 -0.283029 -0.131136  0.053187    -0.032945             -0.698098         -0.053131
299   0.997345  0.996813  0.270609  0.200251 -0.307178  0.873153  0.800368    -0.031731             -0.293160         -0.053854
314   0.247143  0.031195  0.260139  0.261260  0.882335  0.581720  0.979954    -0.032836              1.370903         -0.054448
351   0.594583  0.794384 -0.146733 -0.459772  0.718797  0.397309  0.672754    -0.032361             -0.160616         -0.054206
420   0.997679  0.753711  0.436188  0.621065  0.000000  0.231285  0.527566    -0.031701              1.334138         -0.054272
471   0.901081  0.431261  0.480171  0.046497  0.914528  0.820986  0.839478    -0.032150              0.180663         -0.054101
509  -0.009317  0.998840  0.389204  0.047300  0.000000  0.721618  0.994127    -0.030924              1.934928         -0.054342
527   0.275310  0.955129  0.881384  0.064600  0.875513  0.194844  0.851715    -0.010118             -0.008582         -0.054020
591   0.245619  0.999049  0.300700 -0.059870  0.000000  0.793884  0.786821    34.367195              0.391459         -0.054193
852   0.126200  0.258904  0.146875 -0.178838  0.000000 -0.213063  0.985947    -0.035313              0.914433         -0.054336
892   0.960052  0.940261 -0.180912 -0.197816  0.235786  0.640390  0.423840    -0.032653             -1.025689         -0.043895
966   0.999939  0.423111 -0.182496  0.445861  0.000000 -0.283177  0.672070    -0.032625             -0.714818         -0.053231
986   0.999285  0.998506  0.827404  0.809625  0.956735  0.429413  0.370118    -0.031753             -0.844550         -0.052205
1019  0.970979  0.310574  0.625497 -0.307262  0.000000  0.996675  0.660474    -0.032647              1.488114         -0.054328
1035  0.999970 -0.314415  0.811221  0.064862  0.000000  0.150068  0.974928    -0.030721              1.744308         -0.054266
1077  0.999977  0.621584 -0.593346 -0.419085  0.000000  0.975407  0.704715    -0.031386             -0.887923         -0.051167
```

Figure 9: Outliers

4.3 Verification of outliers using Z-Score and Isolation forest

Both Z-score and Isolation Forest Algorithm was used on the dataset and the outliers were found out. These outlier indices were compared with the outliers found using dimension reduction using VAE and k-means. It was found that most of the indices are common and thus, the method was able to successfully identify outliers.

5 Conclusion

By using a Variational Autoencoder (VAE) for dimensionality reduction, the high-dimensional data was transformed into a compact and meaningful latent representation. Applying K-Means clustering in this lower-dimensional space helped uncover underlying patterns and structures in the data. Outliers were identified by analyzing boundary points in large clusters and detecting small, isolated clusters effectively after this. This approach effectively combined deep learning with traditional clustering technique to enhance anomaly detection, ensuring a robust and scalable solution for identifying outliers in complex datasets.