Assignment 4: GAN Generation of Synthetic Data

Sunayna Padhye   Swarnim Maheshwari   Sankalp Kumar

AI23MTECH12002   CS25MTECH02006   CS24MTECH11026

1 Introduction

Synthetic data generation is an important technique for augmenting datasets, preserving privacy, and enabling robust model development in scenarios where real data is limited or sensitive. Generative Adversarial Networks (GANs) have shown promising results in producing high-fidelity synthetic data, especially in image and tabular data domains.

However, generating tabular data that preserves the marginal distribution (distribution of each feature) and pairwise correlation poses a unique challenge. To address this, we implement a **Wasserstein GAN (WGAN)** combined with **copula-based transformations**. This approach maps tabular features to a Gaussian latent space, enabling stable GAN training while preserving complex statistical properties of the original data.

The goal of this project is to generate synthetic tabular data that closely matches the real data in both marginal distributions and feature correlations. We evaluate the quality of the generated data using statistical metrics such as **Earth Mover's Distance**, **Pearson correlation**, and the **Kolmogorov–Smirnov (KS) test**, along with visual comparisons using KDE plots and correlation heatmaps.

2 Dataset

We use a tabular dataset (`data.xlsx`) with 1199 samples and 10 continuous features: `cov1`–`cov7`, `sal_pur_rat`, `igst_itc_tot_itc_rat`, and `lib_igst_itc_rat`. There are no missing values and all features are numeric. Summary statistics of the raw dataset are shown in table 1. We then apply a *copula transformation* to each feature: map raw values through their empirical CDF into uniform quantiles, apply the inverse standard normal CDF (probit), and clip extremes. This produces a Gaussianized dataset (no longer shown) used as the input for GAN training.

2.1 Data Characteristics

The dataset features are named `cov1` to `cov7`, `sal_pur_rat`, `igst_itc_tot_itc_rat`, and `lib_igst_itc_rat`. Each feature represents a continuous variable.

Basic statistics like mean, standard deviation, and quantiles are shown in Table 1. The features vary considerably in range and distribution, some are concentrated near 1 (e.g., `cov1`, `cov2`) while others have broader or skewed distributions (e.g., `sal_pur_rat`, `lib_igst_itc_rat`).

2.2 Exploratory Data Analysis

To better understand the relationships and distributions within the dataset, we begin by examining the Pearson correlation matrix, shown in Figure 1 (left). This matrix reveals linear relationships between features in the dataset. Strong positive or negative correlations can hint at redundancy or underlying patterns in the data.

Key Observations on Real Data:

- **cov1 and cov2:** Both features are tightly concentrated near 1, with very low standard deviations (0.13 and 0.24 respectively). These follow strongly right-skewed distributions with values mostly in the upper range.

- **cov3 and cov4:** These features show wider distributions and more symmetric shapes, as reflected by higher standard deviations (around 0.4).

- **cov5:** This feature is highly skewed with most values concentrated at 0, which aligns with its 25th, 50th, and 75th percentiles all being zero. A small number of samples extend to positive values, visible in the right tail.

- **cov6 and cov7:** These variables exhibit asymmetric, moderately spread distributions with standard deviations around 0.33–0.38. The density curves show gradual increases toward the upper range.

- **sal_pur_rat and lib_igst_itc_rat:** Both features have long right tails and extreme outliers (max values above 30), while their means are close to zero. These distributions are sharply peaked near their minimum values.

- **igst_itc_tot_itc_rat:** A more balanced and complex distribution with multiple subtle modes and a wide value range from approximately -1.06 to +2.17.

These observations confirm that the dataset includes a mix of tightly packed, symmetric, skewed, and long-tailed features, making it a challenging but meaningful candidate for synthetic tabular data modeling.

Table 1: Summary statistics of raw dataset features in `data.xlsx`

Statistic	cov1	cov2	cov3	cov4	cov5	cov6	cov7	sal_pur_rat	igst_itc_tot_itc_rat	lib_igst_itc_rat
Count	1199	1199	1199	1199	1199	1199	1199	1199	1199	1199
Mean	0.956896	0.855770	0.214263	0.147359	0.036329	0.599809	0.527768	-1.251042e-11	-5.004165e-12	1.918265e-11
Std	0.135031	0.244927	0.408193	0.388080	0.177615	0.334306	0.385322	1.000000	1.000000	1.000000
Min	-0.312219	-0.531958	-0.818128	-0.839158	-0.719622	-0.682734	-0.859529	-3.531330e-02	-1.066436e+00	-5.444774e-02
25%	0.982505	0.840675	-0.095193	-0.143054	0.000000	0.382479	0.245701	-3.284146e-02	-8.884636e-01	-5.424427e-02
50%	0.999235	0.969806	0.175910	0.097584	0.000000	0.691423	0.595623	-3.254101e-02	-3.457085e-01	-5.382146e-02
75%	0.999993	0.996604	0.563061	0.457633	0.000000	0.873218	0.869592	-3.194269e-02	7.059485e-01	-5.191380e-02
Max	1.000000	1.000000	1.000000	0.979015	0.999196	0.999999	1.000000	3.436719e+01	2.177948e+00	3.318828e+01

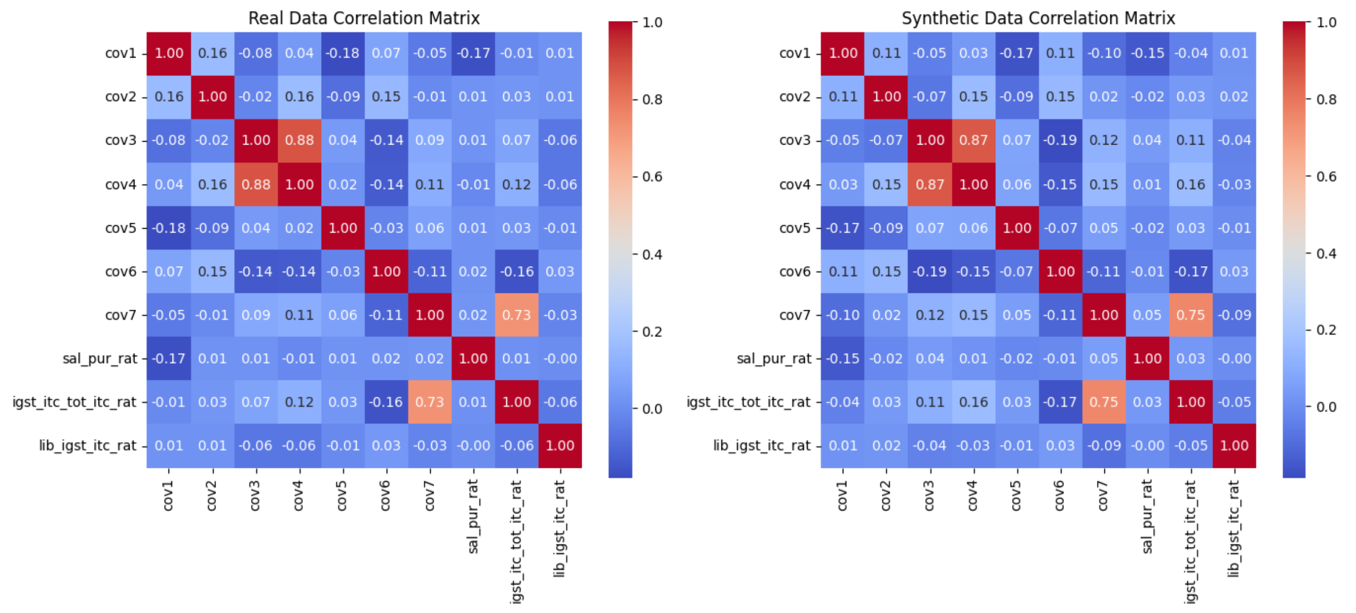

Figure 1: Pearson correlation matrix of real (left) and synthetic (right) dataset features.

3 Methodology

This section outlines the architecture, training process, and overall pipeline used to generate synthetic tabular data using a Copula-based Wasserstein GAN (WGAN-GP).

3.1 Copula Transformation

Tabular features often follow diverse, non-Gaussian distributions. To enable stable GAN training, we apply a copula-based transformation that maps each feature to a standard normal distribution:

- **Forward transform:** Each feature is converted to a Gaussian latent via its empirical cumulative distribution function (CDF), followed by the inverse Gaussian CDF.
- **Inverse transform:** After generation, samples are mapped back to the original data space using the Gaussian CDF followed by inverse empirical interpolation.

This allows the model to learn in a standardized latent space while preserving the original data's marginal distributions.

3.2 Model Architecture

The architecture is composed of two fully connected neural networks:

- **Generator:** Input $\mathcal{N}(0, I)$ vector of dimension 10, two hidden layers of size 128 with BatchNorm + LeakyReLU(0.2), output dimension 10.
- **Critic:** Input 10-dim vector, two hidden layers of size 128 with LeakyReLU(0.2), single-output score.

3.3 Wasserstein GAN with Gradient Penalty (GP)

To improve training stability, we use the WGAN-GP formulation. The critic is updated 5 times for every generator update. The loss function is defined as:

- **Critic loss:** Difference in average critic scores between real and fake samples, plus a gradient penalty term to enforce the Lipschitz constraint.
- **Generator loss:** Negative of the critic's score on fake samples.

The gradient penalty is computed by interpolating between real and fake samples and ensuring the gradient norm is close to 1.

Algorithm 1 WGAN Training & Evaluation

```
Require: Raw data file DATA_PATH, hyperparameters
    {LATENT_DIM, HIDDEN_DIM, LR,
    $\beta_1$, $\beta_2$, $N_{\text{EPOCHS}}$, $B$, $N_{\text{CRITIC}}$, $\lambda_{\text{GP}}$}
 1: Load & preprocess data:
 2:   Read df ← pd.read_excel(DATA_PATH).
 3:   Compute empirical CDFs: $(fwd, inv) \leftarrow$ fit_cdfs(df).
 4:   Gaussianize: $Z_{\text{real}} \leftarrow$ apply_cdf(df, $fwd$).
 5:   Create DataLoader $\mathcal{D}$ from $Z_{\text{real}}$.
 6:
 7: Initialize models & optimizers:
 8:   $G \leftarrow$ Generator(LATENT_DIM, HIDDEN_DIM).
 9:   $D \leftarrow$ Critic(LATENT_DIM, HIDDEN_DIM).
10:   optG ← Adam($G$, LR, $(\beta_1, \beta_2)$).
11:   optD ← Adam($D$, LR, $(\beta_1, \beta_2)$).
12:
13: Training loop:
14: for $e = 1$ to $N_{\text{EPOCHS}}$ do
15:     for each batch $z_{\text{real}} \in \mathcal{D}$ do
16:         real ← $z_{\text{real}}$
17:         for $i = 1$ to $N_{\text{CRITIC}}$ do
18:             Sample $\mathbf{z} \sim \mathcal{N}(0, I)$; fake ← $G(\mathbf{z})$.
19:             Compute gradient-penalty: GP ←
                gradient_penalty($D$, real, fake).
20:             $d\_loss \leftarrow D(\textbf{fake}) - D(\textbf{real}) + \lambda_{\text{GP}} \cdot \text{GP}$.
21:             Update $D$ via optD on $\nabla d\_loss$.
22:         end for
23:         Sample $\mathbf{z} \sim \mathcal{N}(0, I)$.
24:         $g\_loss \leftarrow -D\big(G(\mathbf{z})\big)$.
25:         Update $G$ via optG on $\nabla g\_loss$.
26:     end for
27:     if $e$ mod METRIC_FREQ $= 0$ then
28:         Save checkpoint $G$_epoch $= e$.
29:     end if
30: end for
31:
32: Evaluation:
33: Gather list of saved generator files {G_epoch}.
34: for each checkpoint file do
35:     Load $G_{\text{eval}}$ from file.
36:     syn ← invert_samples($G(z)$, $inv$)
37:     Compute metrics (MSE_corr, Total_EMD) ←
        compute_metrics($G_{\text{eval}}$, fixed_noise, syn, $inv$).
38: end for
39: Normalize and combine scores; select best_epoch.
40: Generate final synthetic data: $\hat{X}$ ←
    invert_samples($G_{\text{best}}$(fixed_noise), $inv$).
41: Save $\hat{X}$ to OUTPUT_PATH.
```

3.4 Training (WGAN-GP)

We train using using Adam (lr=5×10^{-4}, $\beta_1 = 0.5, \beta_2 = 0.9$) for 100,000 epochs. For each epoch:

1. Update critic 5 times: $\mathcal{L}_D = \mathbb{E}[D(\tilde{x})] - \mathbb{E}[D(x)] + 10 \cdot \text{GP}$
2. Update generator once: $\mathcal{L}_G = -\mathbb{E}[D(G(z))]$
3. Save checkpoint at every epoch to `models3/G_epoch{epoch}.pth` for later analysis.

3.5 Pipeline Overview

Algorithm is given in algorithm 1

This structured approach ensures the generated synthetic data captures both marginal feature distributions and inter-feature relationships.

4 Evaluation and Results

We evaluate the performance of the trained CopulaGAN model by comparing the generated synthetic data against the original dataset. We assess quality both statistically and visually, focusing on marginal distributions and inter-feature relationships.

4.1 Evaluation Metrics

Three primary metrics are used to evaluate the synthetic data:

- **Pearson Correlation MSE:** The mean squared error between the off-diagonal elements of the Pearson correlation matrices of real and synthetic data. This measures how well inter-feature dependencies are preserved.
- **Earth Mover's Distance (EMD):** Measures the distributional difference between each real and synthetic feature using optimal transport.
- **Kolmogorov–Smirnov (KS) Test:** A statistical test that compares the cumulative distributions of real and synthetic data. It returns a KS statistic and p-value for each feature. A high p-value (typically > 0.05) suggests that the two distributions are not significantly different.

Each saved generator model is evaluated on the first two metrics, and the one with the lowest combined normalized score (MSE + EMD) is selected as the best generator. The KS test is then used to further validate how well individual feature distributions are captured by the final synthetic dataset.

4.2 Model Selection

All generator checkpoints are loaded and evaluated using a fixed noise input. They are evaluated using a Combined normalized score (EMD + Correlation MSE). The model (corresponding to the epoch) which we get the best score is chosen as the final model.

4.3 Distribution Comparison

We compare real and synthetic distributions for each feature using kernel density estimation (KDE) plots. Figure 2 shows the comparison in the original feature space.

To further validate the shape consistency, we also compare distributions after Z-score normalization, as shown in Figure 3. This helps assess how well the synthetic data aligns in standardized units.

4.4 Earth Mover Distance

We compute the Earth Mover's Distance (EMD) for each feature between the real and synthetic distributions. The results are shown in table 2.

Table 2: Earth Mover's Distance (EMD) between real and synthetic data features.

Covariate	EMD
cov1	0.0003
cov2	0.0007
cov3	0.0005
cov4	0.0004
cov5	0.2132
cov6	0.0011
cov7	0.0009
sal_pur_rat	0.6529
igst_itc_tot_itc_rat	0.0011
lib_igst_itc_rat	0.0146

4.5 Kolmogorov–Smirnov Test Results

The KS test was applied to each feature to statistically compare the real and synthetic distributions. Table 3 summarizes the results.

Table 3: KS test results for real vs. synthetic distributions.

Feature	KS Statistic	p-value
cov1	0.0726	0.0036
cov2	0.0442	0.1919
cov3	0.0325	0.5502
cov4	0.0450	0.1756
cov5	0.4229	1.75×10^{-96}
cov6	0.0267	0.7868
cov7	0.0259	0.8180
sal_pur_rat	0.0425	0.2282
igst_itc_tot_itc_rat	0.0133	0.9999
lib_igst_itc_rat	0.0275	0.7544

Interpretation: Most features (8 out of 10) have high p-values (> 0.05), indicating no significant difference between real and synthetic distributions. However, `cov1` and especially `cov5` exhibit statistically significant differences, which can be seen in distribution plots Figure 2 and not so much after normalization in Figure 3. Notably, `cov5` is highly skewed and sparse, which may explain its divergence.

4.6 Correlation Structure

We evaluate how well the generator preserves feature correlations. Figure 4 shows the absolute element-wise difference between the Pearson correlation matrices of real and synthetic data.

A lower difference across the matrix indicates that the synthetic data maintains the inter-feature dependencies effectively.

5 Conclusion

In this project, we implemented a **Copula-based Wasserstein GAN (CopulaGAN)** to generate realistic synthetic tabular data. The use of a copula transform enabled us to map diverse feature distributions into a Gaussian latent space, facilitating stable and effective GAN training.

Quantitative evaluation using Earth Mover's Distance and Pearson Correlation MSE demonstrated that the generated data **closely matches the real data in both distributional shape and inter-feature dependencies**. Visual analysis using KDE plots confirmed strong alignment across most features, particularly after Z-score normalization.

We further validated the output with the **Kolmogorov–Smirnov test** and **earth mover distance**, which showed a **high degree of statistical similarity** between real and synthetic distributions across the majority of features, reinforcing the quality and realism of the generated data.

In conclusion, CopulaGAN proves to be a robust and interpretable framework for generating high-quality synthetic tabular data—useful in tasks where data sharing, privacy, or augmentation is essential.

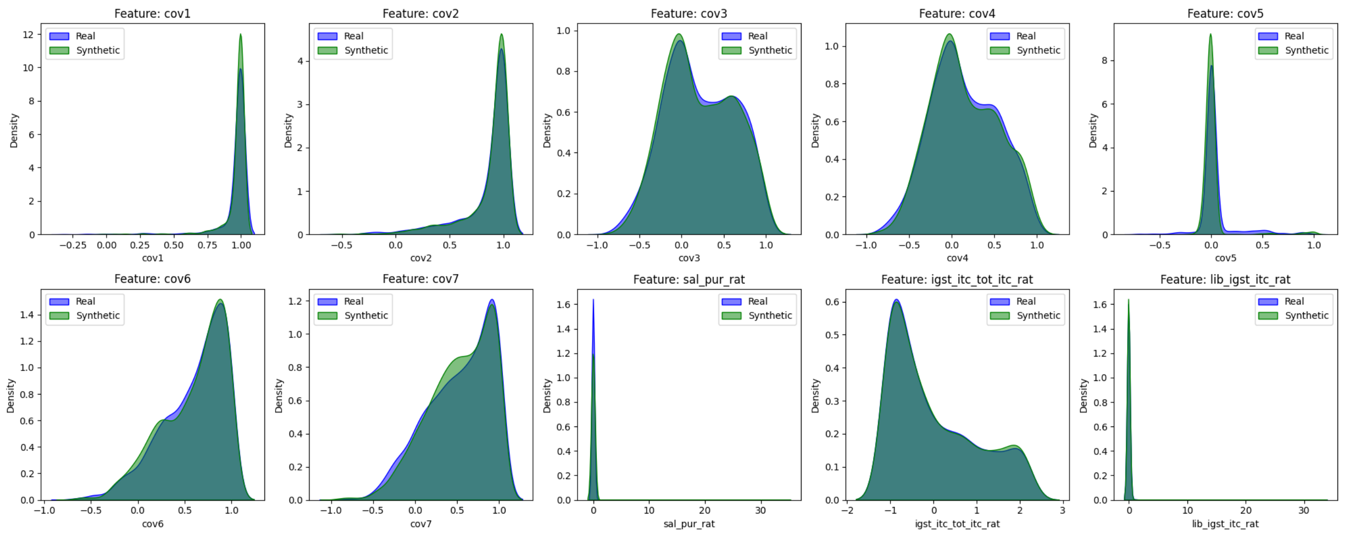

Figure 2: KDE plots comparing real (blue) and synthetic (green) data distributions for all 10 features.

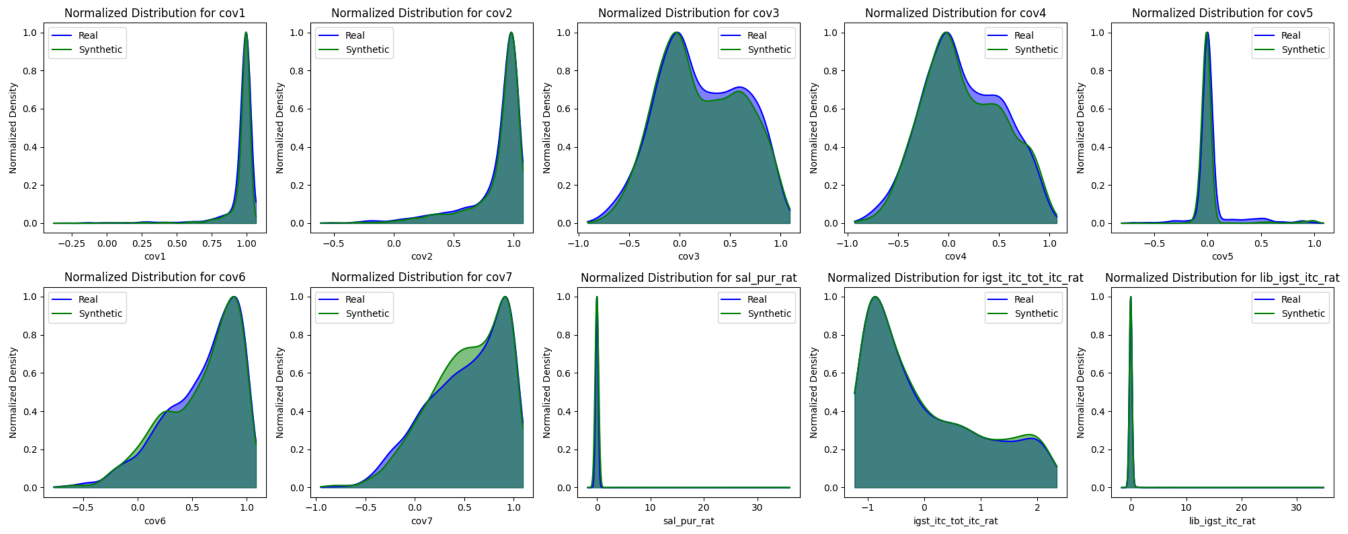

Figure 3: KDE plots comparing real and synthetic distributions on Z-score normalized data.

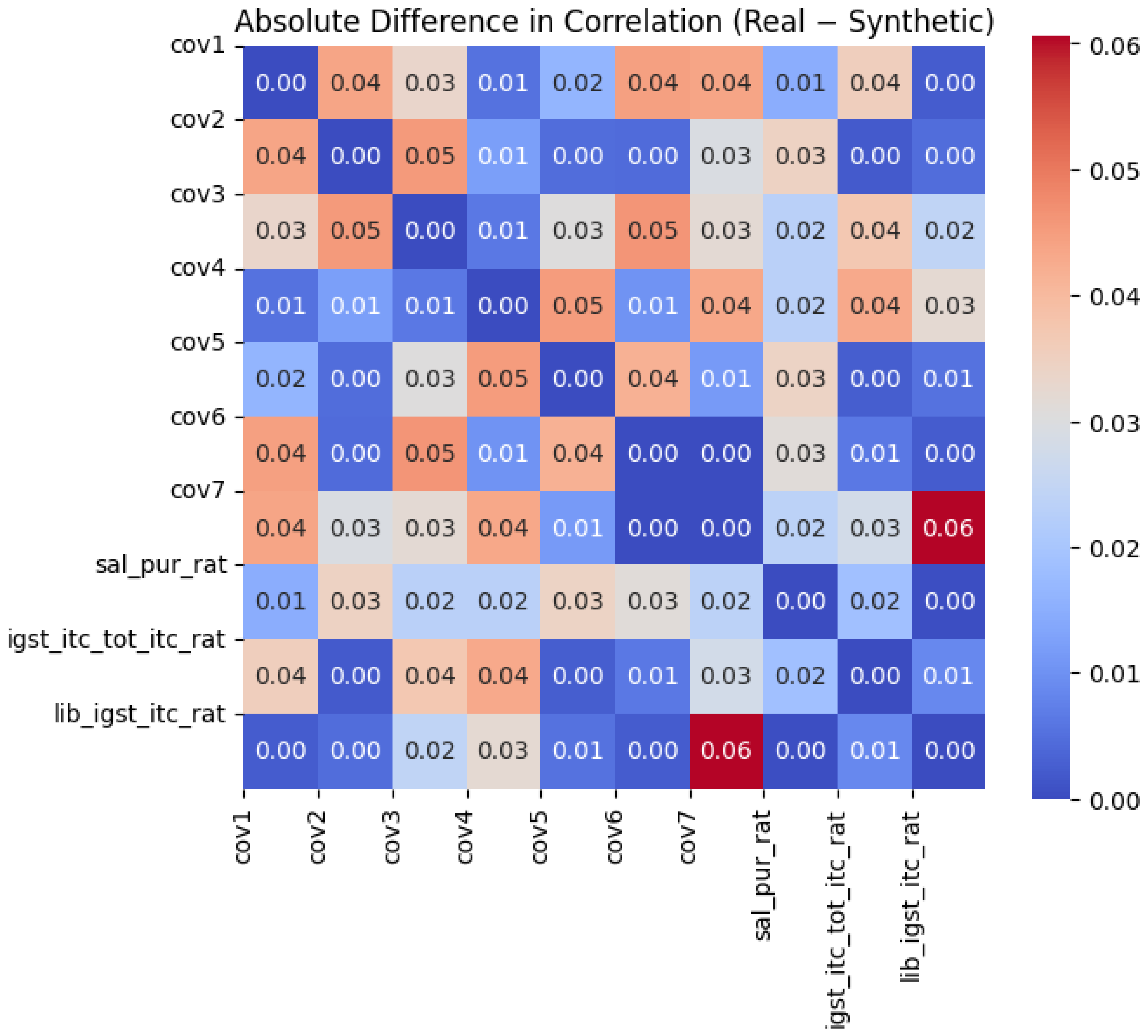

Figure 4: Absolute difference in Pearson correlation between real and synthetic data. Our correlation MSE is approximately equal to 0.000813.