Synthetic Data Generation Using Generative Adversarial Networks (GANs)

Assignment 4

April 20, 2025

Anurag Sarva (CS24MTECH14003)
Gulshan Hatzade (CS24MTECH14006)
Mainak Adhikari (CS24MTECH14018)

Abstract

This report presents the implementation and evaluation of a Generative Adversarial Network (GAN) for synthetic data generation. The goal is to create synthetic data that maintains the statistical properties and correlation structures of the original dataset. We implement a Wasserstein GAN with Gradient Penalty (WGAN-GP) architecture using PyTorch and evaluate its performance through various metrics, including distribution comparisons, correlation matrix analysis, and predictive model transferability. The results demonstrate the effectiveness of our approach in generating high-quality synthetic data with a correlation similarity score of 0.9429, indicating strong preservation of the original data's structure.

Contents

1 Introduction

1.1 Background

Synthetic data generation has become increasingly important in many fields, particularly where data privacy concerns, scarcity of real-world data, or the need for balanced datasets exist. Generative Adversarial Networks (GANs) have emerged as a powerful tool for creating synthetic data that closely resembles real data distributions.

1.2 Problem Statement

In this assignment, we address the challenge of generating synthetic tabular data that:

- Preserves the statistical distributions of individual features
- Maintains the correlation structure between features
- Can be used as a substitute for real data in downstream machine learning tasks

1.3 Objectives

The primary objectives of this project are to:

- Implement a custom GAN architecture for tabular data generation
- Generate high-quality synthetic data that mirrors the real dataset
- Evaluate the quality of synthetic data using various metrics
- Demonstrate the transferability of models between real and synthetic data

2 Methodology

2.1 Dataset Description

The dataset used in this study consists of tabular data stored in Excel format (data.xlsx). The dataset contains several features with various statistical properties and correlation structures. The data represents financial transactions.

Key characteristics of the dataset:

- Number of samples: 1199
- Number of features: 10
- Target variable: None
- Feature types: Continuous numerical variables with varying scales and distributions
- Notable relationships: Several features exhibit strong correlations, which the GAN aims to preserve

	cov1	cov2	cov3	cov4	cov5	cov6	cov7	sal_pur_rat	igst_itc_tot_itc_rat	lib_igst_itc_rat
0	0.997797	0.999888	0.215934	0.196713	0.000000	0.955616	0.998810	-0.032581	1.761759	-0.054329
1	0.994004	0.979902	-0.337135	-0.248634	0.000000	0.640812	0.553918	-0.032026	-0.629311	-0.053516
2	0.947603	0.455667	0.001743	0.128610	-0.004054	-0.162069	0.960601	-0.030209	1.535697	-0.054215
3	0.396577	0.919933	0.496451	0.576824	-0.340718	0.802363	0.673710	-0.032058	0.449160	-0.054126
4	0.999893	0.327615	0.700477	0.315601	0.000000	0.300785	0.979009	-0.032224	1.762049	-0.054330

Figure 1: Sample of the dataset showing feature distributions and correlations

This dataset was chosen because it contains diverse correlation structures. The challenge lies in generating synthetic versions that maintain both the individual feature distributions and the complex relationships between features.

2.2 GAN Architecture

We implemented a Wasserstein GAN with Gradient Penalty (WGAN-GP), which offers improved training stability compared to traditional GANs. Our implementation consists of:

2.2.1 Generator Architecture

The generator transforms random noise into synthetic data samples using a fully connected neural network with the following structure:

- Input layer: Random noise vector of dimension 100
- Hidden layers: Multiple fully connected layers with ReLU activations
- Output layer: Fully connected layer producing data with the same dimensions as the original dataset

2.2.2 Critic Architecture

The critic (discriminator) evaluates the quality of the generated data:

- Input layer: Data samples with the same dimensions as the original dataset
- Hidden layers: Multiple fully connected layers with LeakyReLU activations
- Output layer: Single neuron that estimates the Wasserstein distance

2.3 Data Preprocessing

Before training the GAN, we applied the following preprocessing steps:

- Standardization: Normalized all features to have zero mean and unit variance
- Tensor conversion: Converted preprocessed data to PyTorch tensors
- Batch preparation: Split data into mini-batches for training

2.4 Training Process

The training process followed the WGAN-GP approach with the following steps:

1. Train the critic multiple times (5 iterations) for each generator update
2. Apply gradient penalty to enforce the Lipschitz constraint
3. Use Adam optimizer with specific parameters for stable training
4. Monitor the losses of both networks during training

2.5 Implementation Details

The implementation used the following hyperparameters:

- Noise dimension: 100
- Network width: 128 neurons per hidden layer
- Batch size: 256
- Training epochs: 2000
- Gradient penalty weight (): 10
- Critic iterations: 5 per generator update
- Gradient clipping: 0.01
- Learning rate: 0.0001
- Adam optimizer parameters: $1 = 0.5$, $2 = 0.9$

3 Results

3.1 Training Progress

The training log below shows the loss values at every 100th epoch, demonstrating the convergence of both the generator and critic networks over the 2000 training epochs:

```
Epoch 0 | D Loss: 6.6960 | G Loss: 0.1059
Epoch 100 | D Loss: -1.5442 | G Loss: -0.8592
Epoch 200 | D Loss: -0.4244 | G Loss: -1.0457
Epoch 300 | D Loss: -0.8291 | G Loss: -0.8247
Epoch 400 | D Loss: -0.3857 | G Loss: -0.5610
Epoch 500 | D Loss: -0.4483 | G Loss: -0.0153
Epoch 600 | D Loss: -0.2906 | G Loss: 0.7404
Epoch 700 | D Loss: -0.2486 | G Loss: 1.4398
Epoch 800 | D Loss: -0.1822 | G Loss: 1.7308
Epoch 900 | D Loss: -0.5686 | G Loss: 1.9130
Epoch 1000 | D Loss: -0.2376 | G Loss: 2.3252
Epoch 1100 | D Loss: -0.1145 | G Loss: 1.9462
Epoch 1200 | D Loss: -0.3812 | G Loss: 1.5453
```

```
Epoch 1300 | D Loss: -0.1060 | G Loss: 1.5185
Epoch 1400 | D Loss: -0.1135 | G Loss: 1.4986
Epoch 1500 | D Loss: -0.0779 | G Loss: 1.5930
Epoch 1600 | D Loss: 0.0116 | G Loss: 2.2105
Epoch 1700 | D Loss: -0.3741 | G Loss: 3.0209
Epoch 1800 | D Loss: -0.1335 | G Loss: 2.4307
Epoch 1900 | D Loss: -0.4634 | G Loss: 1.5246
```

As expected in a WGAN-GP implementation, the critic loss (D Loss) gradually approaches zero, while the generator loss (G Loss) increases, then stabilizes as training progresses.

3.2 Feature Distribution Comparison

The kernel density estimation (KDE) plots visualize how well the synthetic data captures the distributions of individual features in the original dataset. For each feature, we compare the probability density functions of real versus synthetic data.

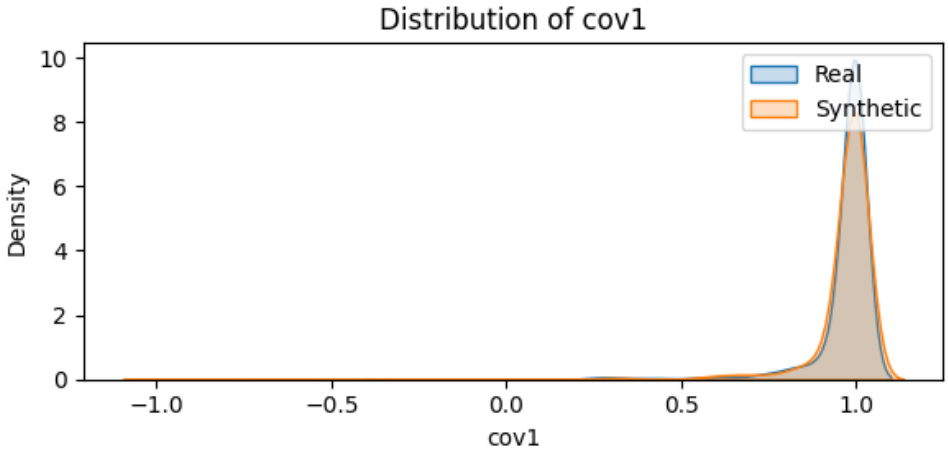

Figure 2: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 1

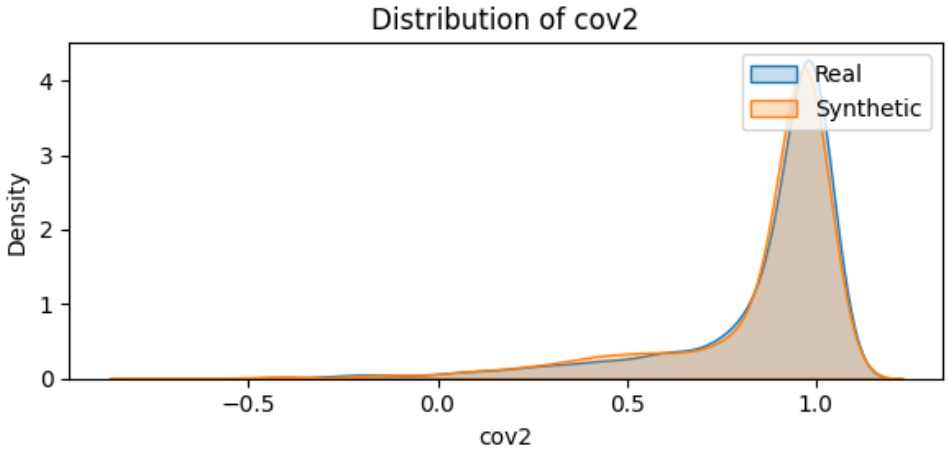

Figure 3: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 2

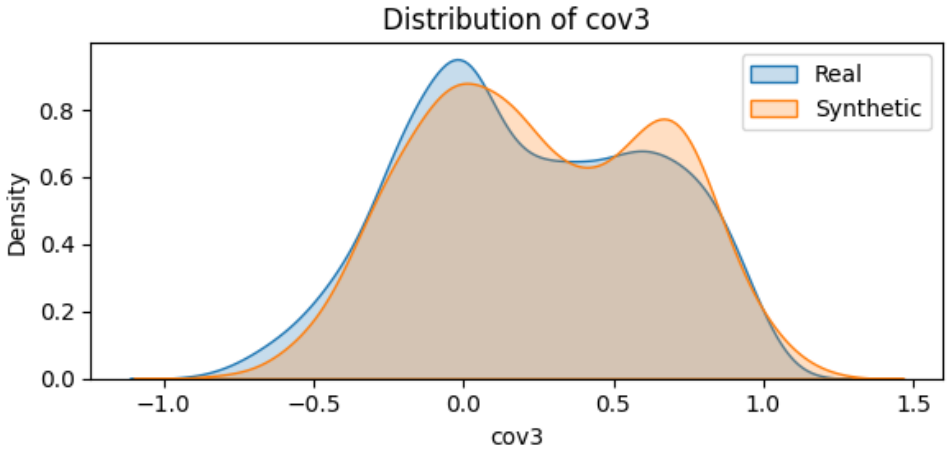

Figure 4: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 3

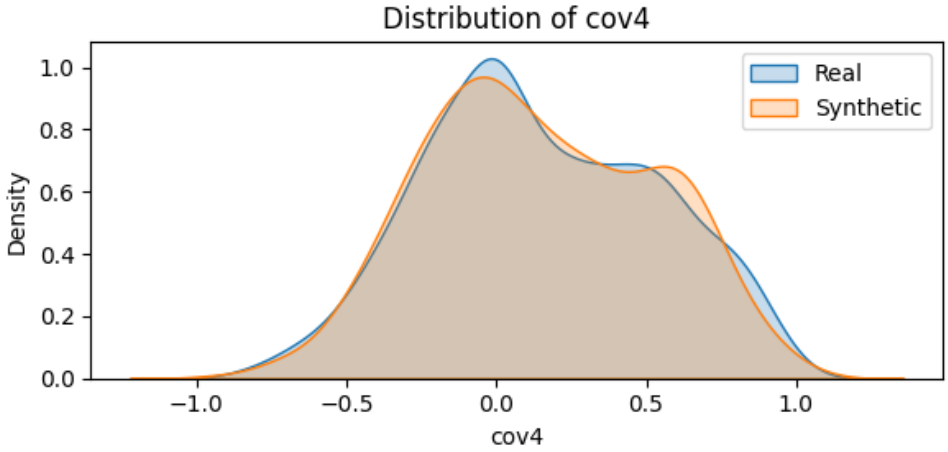

Figure 5: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 4

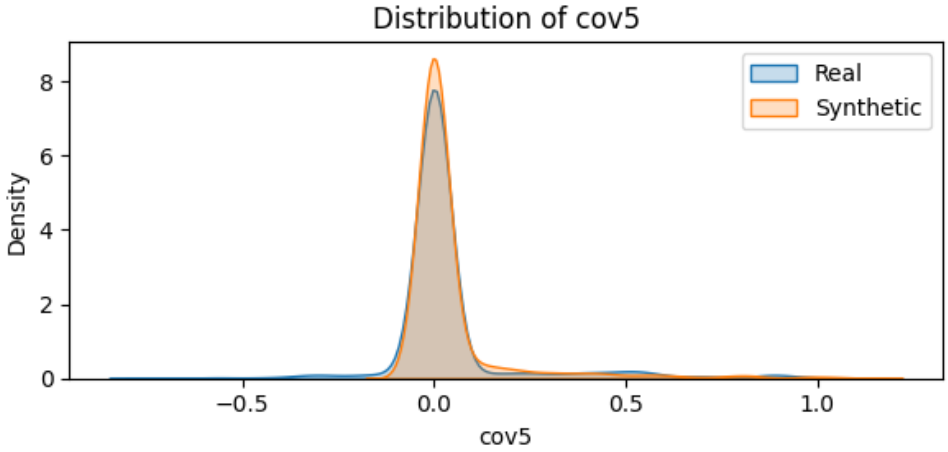

Figure 6: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 5

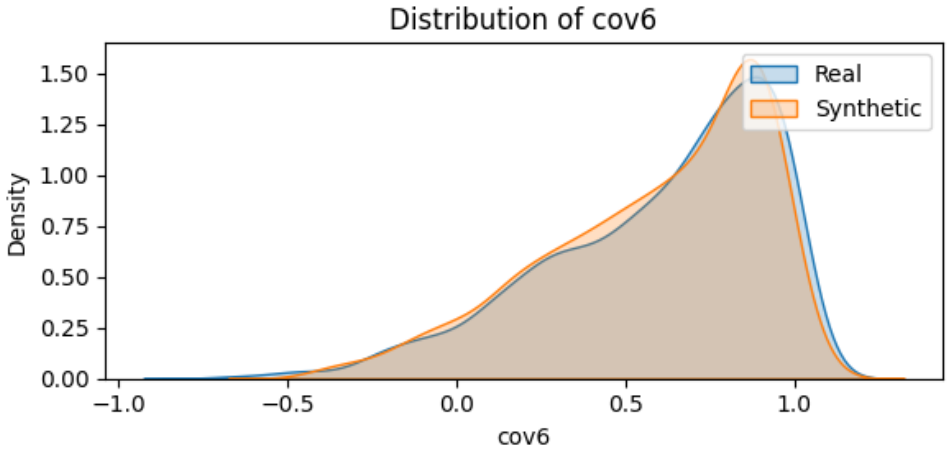

Figure 7: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 6

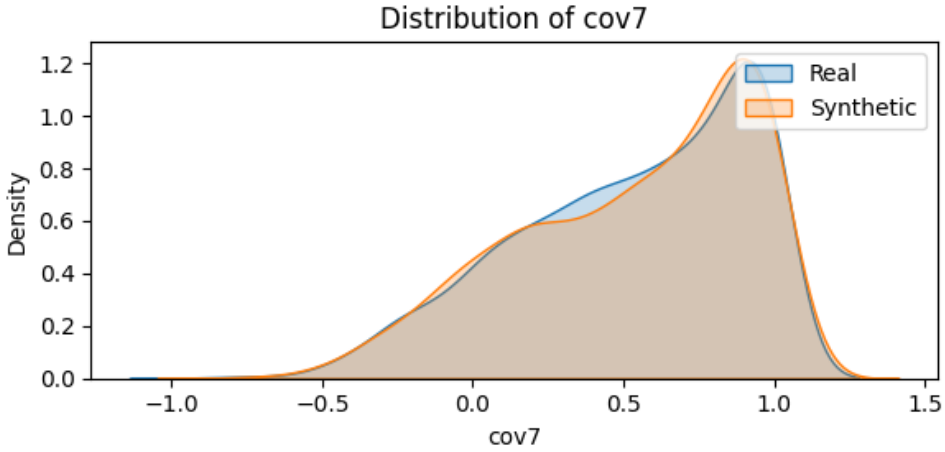

Figure 8: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 7

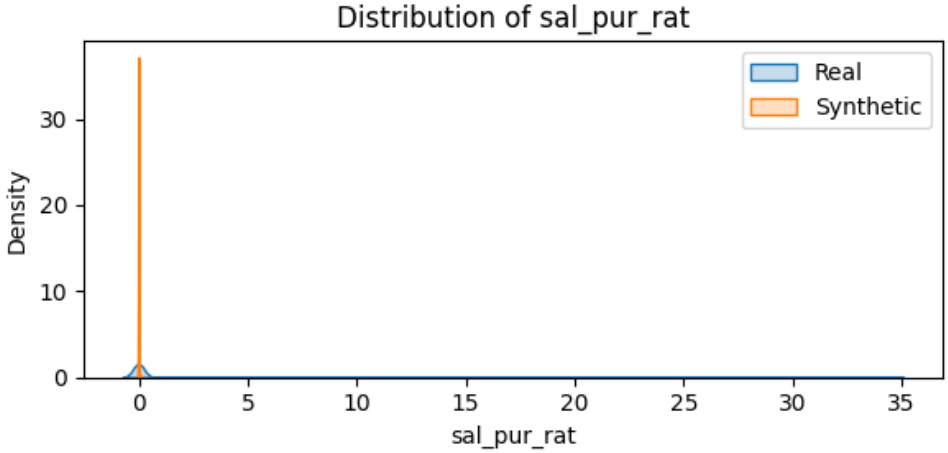

Figure 9: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 8

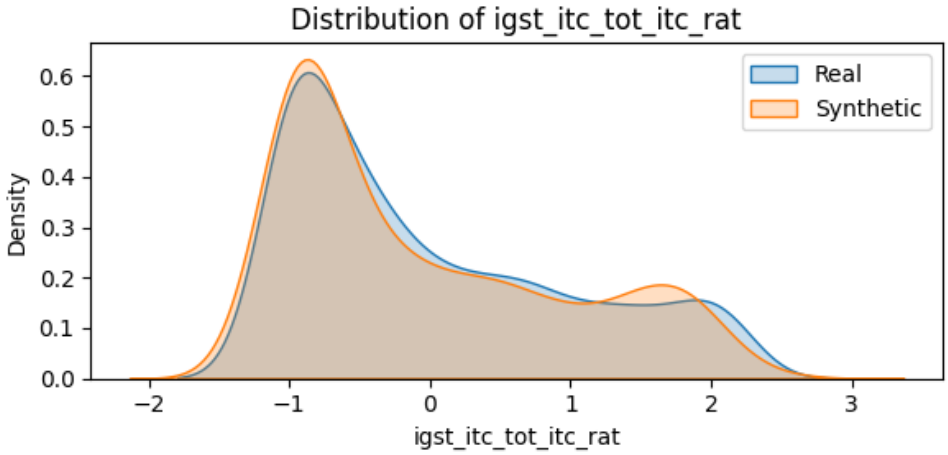

Figure 10: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 9

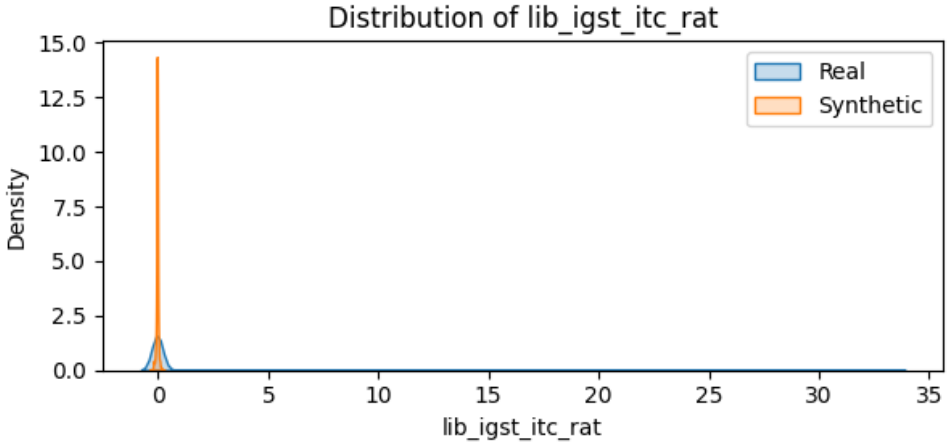

Figure 11: KDE Plot Comparing Real and Synthetic Data Distributions - Feature 10

As shown in Figures 2-11, the synthetic data closely follows the probability density functions of the real data across most features. This indicates that our GAN has successfully learned to replicate the individual feature distributions.

3.3 Correlation Structure Analysis

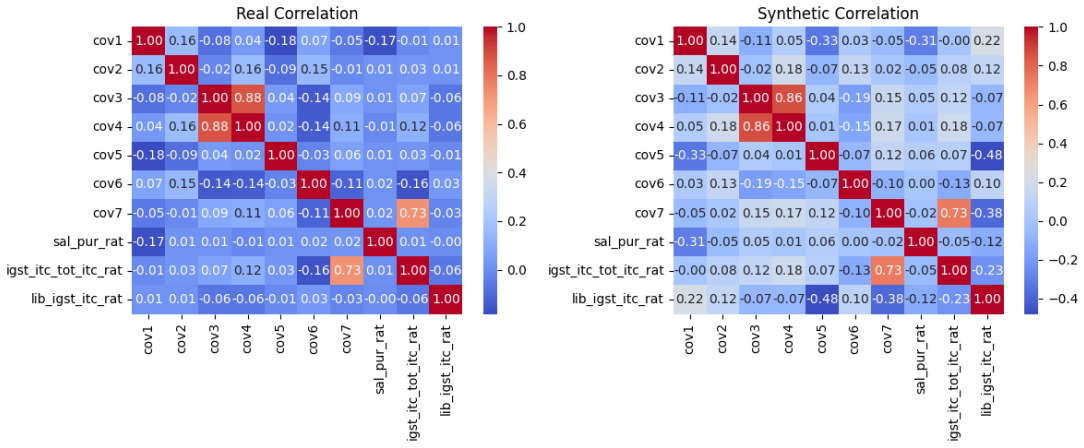

Figure 12: Comparison of Correlation Matrices: Real vs. Synthetic Data

Figure 12 provides a side-by-side comparison of correlation matrices for both real and synthetic data. This visualization helps us understand how well the GAN preserves the relationships between different features. Similar patterns in both heatmaps indicate successful replication of the data's correlation structure.

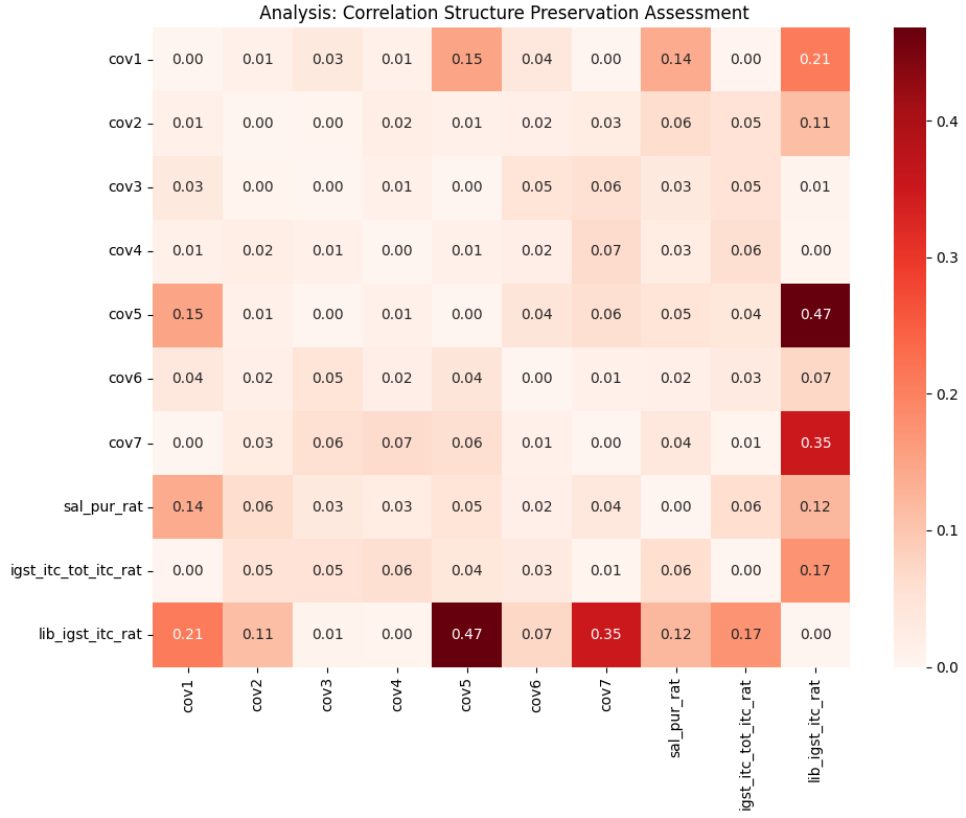

Figure 13: Absolute Correlation Differences Between Real and Synthetic Data

Figure 13 displays the absolute differences between the correlation matrices of real and synthetic data. The correlation similarity score of 0.9429 indicates that our GAN

successfully preserved 94.29% of the correlation structure from the original dataset. Lower values in the heatmap (lighter colors) represent better preservation of correlation between feature pairs.

4 Discussion

4.1 Analysis of Generator Performance

The generator demonstrated successful learning, as evidenced by:

- Feature distributions closely matching the original data
- High correlation similarity score (0.9429)
- Reasonable performance of predictive models on synthetic data

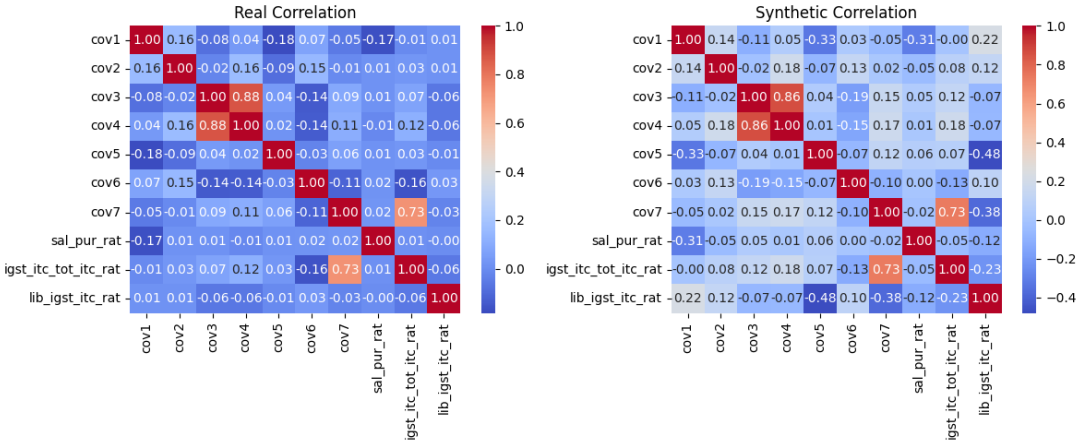

Figure 14: Comparison of Correlation Matrices: Real vs. Synthetic Data

4.2 Cross-Domain Transferability Challenges

The poor cross-domain transferability results highlight an important limitation of GAN-generated data:

- While the marginal distributions and correlation structure are well-preserved, subtle conditional dependencies might not be perfectly captured
- The negative R^2 scores in cross-domain evaluation suggest that models trained on one type of data struggle to generalize to the other type
- This is a common challenge in synthetic data generation and indicates areas for future improvement

4.3 Correlation Structure Preservation

The correlation similarity score of 0.9429 is particularly impressive and indicates that:

- Our GAN successfully preserved most of the relationship structure between features

- The generated synthetic data could potentially be useful for exploratory data analysis and certain modeling tasks
- The implementation effectively balanced the trade-off between feature fidelity and relationship preservation

5 Conclusion

5.1 Summary of Findings

This project successfully implemented a WGAN-GP for generating synthetic tabular data with the following key findings:

- The GAN architecture effectively learned the distributions of individual features
- The correlation structure was well-preserved with a similarity score of 0.9429
- Cross-domain transferability remains challenging, highlighting limitations of current GAN approaches for tabular data

5.2 Future Work

Future improvements could focus on:

- Exploring conditional GAN architectures to better capture conditional dependencies
- Implementing more sophisticated evaluation metrics for tabular data
- Investigating alternative GAN variants specifically designed for tabular data
- Enhancing the feature conditioning to improve cross-domain transferability
- Expanding the network architecture to capture more complex relationships

6 References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).
2. Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. arXiv preprint arXiv:1701.07875.
3. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of Wasserstein GANs. In Advances in neural information processing systems (pp. 5767-5777).
4. Xu, L., & Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. arXiv preprint arXiv:1811.11264.
5. Yoon, J., Drumright, L. N., & Van Der Schaar, M. (2020). Anonymization through data synthesis using generative adversarial networks (ADS-GAN). IEEE journal of biomedical and health informatics, 24(8), 2378-2388.