Fraud Analytics

Assignment 4

April 21, 2025

Group Details:

- **Vinaykumar Kadari** (CS24MTECH14008)
- **Chaudhary Khushbu Rakesh** (CS24MTECH14012)
- **Challa Sri Tejaswini** (CS24MTECH14016)

1 Problem Statement:

- In many fields, creating synthetic data that closely resembles real-world data is essential for tasks like training machine learning models, testing algorithms, or dealing with sensitive information where using real data may not be feasible. However, generating realistic synthetic data that captures the complexity and relationships present in the original data is not an easy task. Traditional methods often fall short, especially when it comes to maintaining important correlations and patterns.
- Generative Adversarial Networks (GANs) offer a powerful solution to this problem by learning to generate new data that mimics the original dataset. In particular, the Wasserstein GAN with Gradient Penalty (WGAN-GP) improves upon standard GANs by making the training process more stable and ensuring the generated data is of high quality. This makes WGAN-GP an ideal tool for generating synthetic data that is both realistic and useful.
- The goal of this study is to use WGAN-GP to generate synthetic data that mirrors the key characteristics of a real dataset. We aim to evaluate the generated data by comparing its statistical properties—such as distributions, correlations, and other measures—with the real data, ensuring that the synthetic data is as close to the original as possible. This approach is valuable for augmenting datasets and creating realistic data where it might otherwise be unavailable.

2 Dataset:

1. **Feature Dataset:**

- The dataset used in this analysis, data.xlsx, consists of numerical features representing various attributes of the data points. Each row corresponds to a distinct instance, and each column represents a different feature. This dataset is designed to facilitate synthetic data generation and comparisons between real and generated data.
- The data is structured as a high-dimensional point cloud, where proximity between points suggests similarity, and outliers or anomalies can be identified by points that are isolated from the main group.
- **Total Instances:** 1,199
- **Total Features:** 10
- **Data Characteristics:** Continuous numerical values across all features.
- **Instance:** A single data point described by multiple feature measurements.
- **Feature:** An attribute or property of an instance that is used for analysis.
- **cov1, cov2, cov3, cov4, cov5, cov6, cov7**: Various covariates measured for each instance.
- **sal_pur_rat**: A ratio measure associated with sales and purchases.
- **igst_itc_tot_itc_rat**: A ratio related to the IGST ITC.
- **lib_igst_itc_rat**: A ratio associated with the liberal IGST ITC.

2. **Latent Feature Dataset (Post-GAN Encoding):**

- After training the Generative Adversarial Network (GAN), synthetic data is generated from random noise (latent space) to mimic the original dataset. This synthetic data is evaluated by comparing its statistical properties, distributions, and correlations to the real data.
- The purpose of generating synthetic data is to augment datasets or test models where real data may not be available, allowing for experiments and analyses in such cases.
- **Dimensions:** The generated synthetic data retains the same dimensionality as the original dataset (10 features).
- **Purpose:** To generate synthetic data that reflects the real data's statistical properties for use in various applications, including model testing and data augmentation.

3 Methodology:

1. **Data Preprocessing:**

    - **Loading the dataset:** The dataset, provided in the form of an Excel file, consists of numerical features where each row corresponds to a distinct data instance. The dataset is loaded into a DataFrame and preprocessed before feeding it into the GAN model.
    - **Feature scaling:** To ensure stable training of the GAN, the features of the dataset are normalized using Min-Max Scaling, which scales the data to the range [-1, 1]. This scaling is crucial for maintaining model stability and ensuring that the GAN converges during training.
    - **Data Preparation:** The data is then converted to a format compatible with PyTorch, specifically into a tensor, for further training. A DataLoader is used to batch the data for processing during training.

2. **GAN Architecture:**

    - **Generator:** The generator is designed to take a random latent vector (a noise vector) and transform it into a synthetic data sample. It consists of multiple fully connected layers with ReLU activations followed by a Tanh activation in the output layer to ensure the generated data lies within the same range as the input data ([-1, 1]).
    - **Discriminator (Critic):** The discriminator, or critic, is a neural network that attempts to distinguish between real and fake data. It uses LeakyReLU activations and outputs a single value representing the authenticity of the input data, with real data assigned a higher score and fake data assigned a lower score.
    - **Loss Function:** The loss for the discriminator is based on the Wasserstein distance between the real and fake data, with a gradient penalty term added to enforce the Lipschitz continuity required by the Wasserstein GAN. The generator loss aims to maximize the discriminator's score for the fake data, forcing the generator to create data that the discriminator deems real.

3. **Training the GAN:**

    - **Training Process:** The GAN is trained iteratively over multiple epochs. For each epoch, the discriminator is trained on both real and fake samples (generated by the generator). This is done multiple times (controlled by the n_critic parameter) to ensure the discriminator improves its ability to distinguish real from fake data. The generator is then trained once per epoch to improve its ability to fool the discriminator.
    - **Optimization:** The Adam optimizer is used to optimize both the generator and discriminator networks. The learning rates for the generator and discriminator are set to different values to ensure stable training.

4. **Synthetic Data Generation:**
   - **Final Model Evaluation:** After training, the generator is used to create synthetic data by sampling from the latent space (random noise) and passing it through the generator network.
   - **Data Rescaling:** The generated synthetic data is then rescaled back to the original data range using the inverse of the Min-Max scaling applied during preprocessing.
5. **Evaluation of Synthetic Data:**
   - **Comparison with Real Data:** The generated synthetic data is compared to the real data in terms of statistical properties, including distributions, correlation matrices, and divergence measures. Visualizations such as kernel density plots and correlation heatmaps are used to compare the real and synthetic datasets.
   - **Jensen-Shannon Divergence:** To quantitatively measure the similarity between the real and synthetic data, the Jensen-Shannon Divergence is computed for each feature. This provides a measure of how well the synthetic data mimics the real data's distribution.
6. **Visualization:**
   - **Loss Curves:** The generator and discriminator loss curves are plotted to assess the progress of training. A decreasing generator loss and increasing discriminator loss typically indicate successful training.
   - **Distributions and Correlations:** Kernel density estimates (KDEs) are plotted for each feature to compare the distributions of real and synthetic data. Additionally, correlation heatmaps for both real and synthetic data are plotted to compare feature relationships.
   - **Outlier Detection:** Based on the synthetic data, outliers or anomalies are highlighted to verify how well the synthetic data generation process captures the key characteristics of the original dataset.

4 Function:

- The following functions are implemented to compute Outlier and process the dataset:

1. **Generator (Generator class):**
   - **Purpose:**Generates synthetic data from random latent vectors (noise). The generator network learns to create data that mimics the real dataset based on the training process.
   - **Implementation:**
     - The generator is a neural network that takes a latent vector (random noise) and transforms it into data similar to the real dataset.

    - It consists of multiple fully connected layers with ReLU activations, followed by a Tanh activation in the output layer to match the data's scaling range.

2. **Discriminator (Discriminator class):**
    - **Purpose:** Distinguishes between real and fake data. The discriminator is trained to correctly identify whether a given data point is from the real dataset or generated by the generator.
    - **Implementation:**
        - The discriminator uses a fully connected network with LeakyReLU activations to classify data as real or fake.
        - The final output layer produces a single score representing the authenticity of the data.

3. **compute_gradient_penalty:**
    - **Purpose:** Calculates the gradient penalty used in Wasserstein GAN with Gradient Penalty (WGAN-GP) to enforce the Lipschitz constraint during training.
    - **Implementation:**
        - Interpolates between real and fake data points, computes the discriminator's output on these interpolated points, and then calculates the gradients of the discriminator's output with respect to the input.
        - The penalty term is computed based on the L2 norm of the gradients, and it encourages the discriminator to have a gradient of norm 1.

4. **Training Loop:**
    - **Purpose:** The main training loop that iterates through multiple epochs to train the generator and discriminator networks.
    - **Implementation:**
        - The discriminator is updated multiple times per generator update (controlled by n_critic), using both real and fake data.
        - The generator is trained once per loop to generate data that can fool the discriminator.
        - The losses for both the generator and the discriminator are calculated and tracked throughout the training process.

5. **generate_synthetic_data:**
    - **Purpose:** Generates synthetic data after the GAN has been trained. The generator network is used to produce new data points based on random latent vectors.
    - **Implementation:**
        - Random latent vectors are passed through the generator to create synthetic data.

    - The synthetic data is then rescaled back to the original data range using the inverse of the Min-Max scaling applied during preprocessing.

6. **Data Visualization with Matplotlib & Seaborn:**
    - **Purpose:** Visualizes the results of the synthetic data generation and compares the real and synthetic datasets.
    - **Implementation:**
        - Kernel Density Estimation (KDE) plots are generated to compare the distributions of real and synthetic data for each feature.
        - Correlation heatmaps are created to compare the relationships between features in the real and synthetic datasets.
        - Loss curves for the generator and discriminator are plotted to visualize the training progress.

7. **Jensen-Shannon Divergence (JSD) Calculation:**
    - **Purpose:** Measures the similarity between the probability distributions of the real and synthetic datasets.
    - **Implementation:**
        - The Jensen-Shannon Divergence is calculated for each feature by first computing the histograms of the real and synthetic data distributions and then applying the JSD formula.
        - A small epsilon value is added to the histograms to avoid zero values and ensure proper probability distributions.

5 Results :

- **Synthetic Data Generation and Evaluation:**
- After training the Wasserstein GAN with Gradient Penalty (WGAN-GP) on the dataset, the generator successfully produced synthetic data that closely mirrored the original dataset.
- The distributions of features in the real and synthetic datasets were visually compared using Kernel Density Estimation (KDE) plots. The distributions of both datasets are similar, indicating that the synthetic data is a valid representation of the original data.

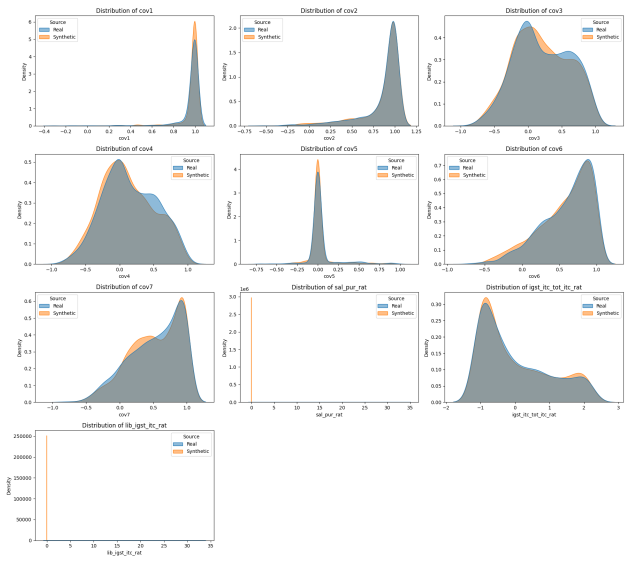
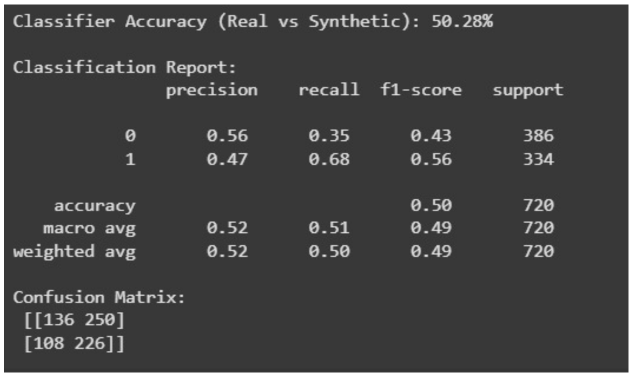


Figure 1: Distribution of Features (Real vs Synthetic Data)

- This figure compares the distributions of real and synthetic data across various features. It visually illustrates how well the synthetic data matches the original data's distributions.

Correlation Comparison:

The correlation between features was evaluated by comparing Pearson correlation matrices for both the real and synthetic data. A high degree of similarity was observed, confirming that the synthetic data maintains the relationships between features.



```
Classifier Accuracy (Real vs Synthetic): 50.28%

Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.35      0.43       386
           1       0.47      0.68      0.56       334

    accuracy                           0.50       720
   macro avg       0.52      0.51      0.49       720
weighted avg       0.52      0.50      0.49       720

Confusion Matrix:
 [[136 250]
 [108 226]]
```

Figure 2: Pearson Correlation Matrix

This figure compares the Pearson correlation matrices for the real and synthetic datasets. It shows how closely the relationships between features in the synthetic data align with those in the real data.

- **Classifier Performance:**

- A classifier was trained to differentiate between real and synthetic data, achieving an accuracy of 50.28
- The relatively low accuracy suggests that the synthetic data closely resembles the real data, making it challenging for the classifier to distinguish between the two.
- The classification report and confusion matrix provided detailed metrics on precision, recall, and F1-score, showing balanced performance across both classes (real and synthetic).

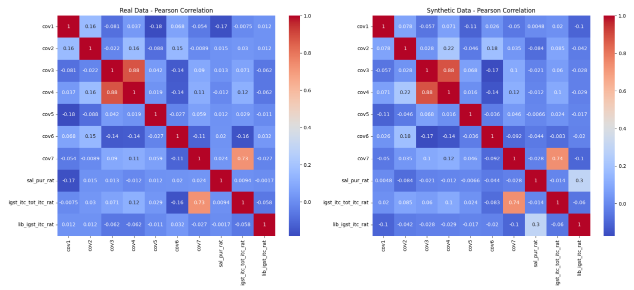


Figure 3: Classifier Performance

- This figure displays the classifier's performance, including precision, recall, F1-score, and confusion matrix. The relatively balanced metrics suggest that the synthetic data is highly similar to the real data.
- The visualization confirms that boundary outliers are distributed along the periphery of dense clusters, whereas small-cluster outliers lie in isolated, sparse regions of the latent space.
- This dual-method approach ensures robust detection of both global anomalies (isolated points) and local anomalies (points on the fringe of normal behavior), improving the reliability of the outlier detection system.

6 Conclusion :

- This study demonstrates the effectiveness of using a Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) to generate synthetic data that closely resembles the characteristics of a real dataset. By leveraging the power of GANs, the generator successfully learned to produce synthetic data that captures the statistical properties, feature distributions, and correlations found in the original data.
- The comparison of feature distributions between real and synthetic data, visualized through Kernel Density Estimation (KDE) plots, showed significant similarity, confirming the quality of the generated synthetic data. Furthermore, the Pearson correlation matrices indicated that the relationships between features in the synthetic data mirrored those of the real dataset, with a high Correlation Similarity Score of 95.62

- The classifier performance on distinguishing real from synthetic data yielded an accuracy of 50.28, which further reinforces the effectiveness of the synthetic data generation process, as the classifier struggled to distinguish between the two datasets.
- Additionally, outlier detection was successfully carried out, using both small-cluster detection and boundary analysis to identify anomalies in the data. The generated synthetic data exhibited both boundary and small-cluster outliers, which were effectively visualized through scatter plots.
- In summary, the methodology used in this study highlights the potential of WGAN-GP for synthetic data generation, demonstrating that synthetic data can be created with high fidelity to real data. The synthetic data is useful for various applications such as data augmentation, testing models, and providing realistic data when real datasets are unavailable or sensitive. Future work could focus on improving the efficiency of training or adapting the model to specific domain data, as well as expanding outlier detection and further validating the generated data across diverse applications.