Final Project Report
- CTGAN Fraud Detection

Project Title: Enhancing Fraud Detection using Synthetic Transactions (CTGAN)
Intern: Shraddha Rajkumar Dodke
Company: GLOBAL NEXT CONSULTING INDIA PRIVATE LIMITED (GNCIPL)
Duration: 6 Weeks

1. Abstract

Fraud detection in financial transactions is challenging due to highly imbalanced datasets. This project enhances fraud detection by generating synthetic transaction data using CTGAN (Conditional Tabular GAN). The synthetic data helps balance classes, improving classifier performance while maintaining data privacy.

2. Objective

- Generate synthetic transactions using CTGAN to handle class imbalance.
- Improve the accuracy and recall of fraud detection models.
- Provide a framework for secure, privacy-preserving data augmentation.

3. Scope

- Applicable to credit card and banking transaction datasets.
- Useful for financial institutions to detect rare fraudulent transactions.
- Can extend to anomaly detection in other tabular datasets.

4. Tools & Technologies

- Python (Pandas, Numpy, Scikit-learn)
- CTGAN (from SDV library)
- Machine Learning Models: Random Forest, XGBoost, Logistic Regression
- Jupyter Notebook / VS Code
- GitHub for source code management
- Streamlit for testing predictions (optional demo)

5. Methodology:

5.1 Data Collection

- Dataset: creditcard.csv (284,807 transactions; 31 features)
- Features include transaction amount, timestamp, anonymized user features, and class label (fraud/non-fraud).

5.2 Data Preprocessing

- Missing value handling
- Feature scaling (StandardScaler/MinMaxScaler)
- Train-test split (70:30)

5.3 Synthetic Data Generation

- CTGAN generates realistic fraudulent transactions conditioned on original data distribution.
- Augmented dataset ensures better class balance (Fraud:Non-Fraud ratio improved).

5.4 Model Training

- Trained models on both original and augmented datasets.

- Hyperparameter tuning using GridSearchCV for best performance.

5.5 Evaluation Metrics

- Precision, Recall, F1-Score, ROC-AUC
- Confusion Matrix to visualize class-wise performance.

6. System Design

6.1 Architecture Diagram

Original Dataset --> Data Preprocessing --> CTGAN
--> Augmented Dataset
--> ML Model Training & Testing
-->Predictions

6.2 UML / Flow

- Data Flow Diagram (DFD)
- Model Training Workflow
- Prediction & Evaluation

7. Implementation

- Step 1: Load and preprocess dataset
- Step 2: Train CTGAN to generate synthetic fraud transactions
- Step 3: Augment original dataset
- Step 4: Train ML models on augmented data
- Step 5: Evaluate models
- Step 6: Deploy using Streamlit (optional demo)

8. Results

Model	Original Data	Augmented Data	Improvement
Random Forest	F1: 0.87	F1: 0.99	+12%
XGBoost	F1: 0.82	F1: 0.99	+17%
Logistic Reg.	F1: 0.65	F1: 0.78	+13%

- ROC-AUC improved from 0.936 → 0.994

- XGBoost: similar improvement observed
- Confusion matrix shows better detection of rare fraud transactions
- Recall for fraud class increased from 0.78 → 0.98 (Random Forest).

9. Testing

- Unit Tests: Verified preprocessing functions, CTGAN output, model pipelines
- Integration Tests: Ensured data flows correctly from synthetic generation to model training
- System Tests: Evaluated overall fraud detection accuracy

10. Deployment Guide

1. Install Python 3.9+
2. Install required libraries: pip install pandas numpy scikit-learn ctgan streamlit
3. Load trained model (joblib/pickle)

4. Run Streamlit app: streamlit run app.py
5. Upload CSV → Predict Fraud

11. User Manual

- Upload transaction CSV in the same format as creditcard.csv
- Click Predict
- View results with probability of fraud and prediction class

12. Conclusion

- CTGAN-based data augmentation improves fraud detection accuracy
- Helps handle data imbalance without compromising privacy
- Framework can extend to other tabular anomaly detection tasks.

13. References

1. Xu, L., et al., "Modeling Tabular Data using Conditional GAN", 2019.
2. Kaggle Credit Card Fraud Detection Dataset: creditcard.csvScikit-learn documentation: https://scikit-learn.org/
3. SDV / CTGAN: https://sdv.dev/SDV/

14. Annexures

- Weekly Progress Reports
- Test Cases & Results
- Source Code (GitHub)
- Declaration & Acknowledgement