National College of Ireland

Configuration Manual

MSc Research Project
MSc. in Data Analytics

Jisoo Park
Student ID: 23376589

School of Computing
National College of Ireland

Supervisor: Shubham Subhnil

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Jisoo Park

Student ID: 23376589

Programme: MSc in Data Analytics **Year:** 2024/2025

Module: MSc Research Practicum

Supervisor: Shubham Subhnil

Submission Due Date: 11/08/2025

Project Title: A Comparative Study of Generative Oversampling Methods for Imbalanced Fraud Classification

Word Count: 1305 **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jisoo Park

Date: 2025/08/10

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jisoo Park
Student ID: 23376589

1 Project Overview

This project addresses the extreme class imbalance in credit card fraud detection (fraud ratio < 0.2%) by comparing traditional oversampling methods with modern generative approaches. An Enhanced Conditional GAN (ECGAN) is proposed, integrating Feature Matching Loss, Gradient Penalty, and Conditional Generation to produce classifier-aware synthetic fraud samples.

Experiments use the public Credit Card Fraud Detection dataset. The comparison includes:

- Traditional: SMOTE, ADASYN, RandomOverSampler, SMOTE-Tomek
- Generative: CTGAN, TVAE, Gaussian Copula
- Proposed: ECGAN

Code is implemented in Python 3.12 on Google Colab. Key packages include pandas, numpy, scikit-learn, imbalanced-learn, torch, and sdv (see Section 3 for full list).

Processing steps:

- Load and preprocess data (scaling, transformation, stratified split)
- Apply oversampling/generative method
- Train XGBoost classifier
- Evaluate with precision, recall, F1, ROC-AUC, PR-AUC, and statistical tests

Detailed configurations, results, and visualizations follow in later sections.

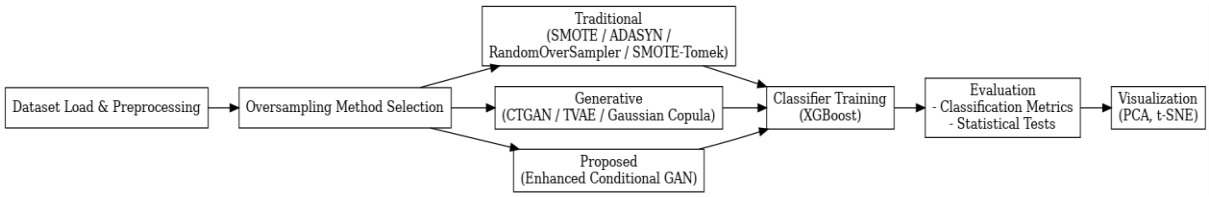

Figure 1: Project Flow.

2 Files and Descriptions

Table 1: Project Files and Descriptions

File Name	Description
creditcard.csv	Original credit card transaction dataset
enhanced_cgan.ipynb	Enhanced Conditional GAN training code
ctgan_model.ipynb	CTGAN training and oversampling code

tvae_model.ipynb	TVAE training and oversampling code
gaussian_copula.ipynb	Gaussian Copula-based oversampling code
smote_adasyn.ipynb	Traditional oversampling code (SMOTE, ADASYN, etc.)
evaluation_metrics.py	Evaluation metrics calculation functions
visualization_plots.ipynb	PCA and t-SNE visualization code
statistical_tests.py	KS-test, JSD, correlation similarity calculation code

3 Tools and Environment Setup

- **Programming Language:** Python 3.12
- **IDE:** Google Colab
- **Execution Environment:** Local Machine, CPU execution
- **Libraries Used:**

Table 2: Used Libraries, Versions and Purpose

Library	Version	Purpose	Documentation Link
pandas	2.2.2	Data processing	https://pandas.pydata.org/docs/
numpy	1.24.4	Numerical computation	https://numpy.org/doc/
scikit-learn	1.6.1	Preprocessing, modeling, evaluation metrics	https://scikit-learn.org/stable/documentation.html
imbalanced-learn	0.13.0	Traditional oversampling methods	https://imbalanced-learn.org/stable/
torch	2.6.0	Deep learning, GAN implementation	https://pytorch.org/docs/stable/index.html
sdv	1.4.0	Synthetic data generation (CTGAN, etc.)	https://docs.sdv.dev/
xgboost	3.0.3	Classifier training	https://xgboost.readthedocs.io/en/stable/
scipy	1.15.3	Statistical tests, distance calculation	https://docs.scipy.org/doc/scipy/

4 Datasets

4.1 Credit Card Fraud Dataset

- **Source:** Public credit card transaction dataset (Dal Pozzolo et al., 2015) (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)
- **Total Samples:** 284,807 transactions
- **Fraud Cases:** 492 (≈ 0.17%)
- **Imbalance Ratio:** About 1:577

- **Features:** 30 columns (Time, Amount, PCA-transformed features V1~V28, Class)
- **Target Variable:** Class (1 = Fraud, 0 = Non-Fraud)

4.2 Preparation

- **EDA (Exploratory Data Analysis)**
  - **Amount feature log-transformed, time patterns preserved**
  - Visualized fraud vs. non-fraud distributions (histograms, KDE plots)
- **Feature Scaling:** Applied StandardScaler to all features
- **Data Split**
  - Stratified Train/Test Split (maintain class ratio)
  - Train: model training and oversampling applied
  - Test: performance evaluation (no oversampling applied)
- **File Format:** CSV file (creditcard.csv)
- **Use in Experiments:**
  - Evaluation of traditional oversampling methods
  - Training and evaluation of generative oversampling methods

```
df = pd.read_csv("/content/credit_default.csv")
assert df.isnull().sum().sum() == 0, "Missing values detected"
total_samples = len(df)
fraud_cases = df[df['Class'] == 1].shape[0]
imbalance_ratio = fraud_cases / total_samples

print(f"Total Samples: {total_samples}")
print(f"Fraud Cases: {fraud_cases} ({imbalance_ratio*100:.4f}%)")
print(f"Imbalance Ratio: 1:{(total_samples - fraud_cases) // fraud_cases}")

# Split features and target
X = df.drop("Class", axis=1)
y = df["Class"]
# Log-transform and normalize
X["Amount"] = np.log1p(X["Amount"])
X["Time"] = np.log1p(X["Time"])
scaler = MinMaxScaler()
X[["Time", "Amount"]] = scaler.fit_transform(X[["Time", "Amount"]])
```

Figure 2: Dataset EDA.

5 Baseline t-SNE Implementation

- **No Oversampling**: Uses the original dataset without applying any oversampling method.
- **Purpose**: Serves as the baseline for comparison with traditional and generative oversampling methods.
- **Classifier**: XGBoost (same hyperparameters as all experiments)
  - XGBoost baseline: Default parameters tuned for imbalanced classification

6 Model Configuration

6.1 Traditional Oversampling Methods

These methods synthetically increase the minority (fraud) class using interpolation or resampling:

- **RandomOverSampler** : Duplicates minority sample (sampling_strategy='minority').

- **SMOTE** : Generates new samples via nearest-neighbor interpolation (k_neighbors=5).
- **ADASYN** : Similar to SMOTE but focuses on harder-to-learn samples (n_neighbors=5).
- **SMOTE-Tomek** : Combines SMOTE with Tomek Links removal to reduce class overlap.
- **Hyperparameters** : k_neighbors=5, sampling_strategy='minority'
- **Key Features** : Simple and computationally efficient, but may produce limited diversity in high-dimensional data.

6.2 Generative Models

Probabilistic and deep learning–based models to learn the joint data distribution and generate synthetic samples:

- **CTGAN** : Conditional Tabular GAN (batch=500, epochs=300) with fully connected layers and Batch Normalization for mixed-type tabular data.
- **TVAE** : Tabular Variational Autoencoder (batch=500, epochs=300) using MLP-based encoder–decoder to learn latent representations.
- **Gaussian Copula** : Models feature dependencies using copula functions with default parameters.
- **Hyperparameters**
    - **CTGAN**: batch=500, epochs=300
    - **TVAE**: batch=500, epochs=300
    - **Gaussian Copula**: Default parameters
- **Key Features**
    - CTGAN and TVAE capture complex nonlinear dependencies.
    - Gaussian Copula is effective at preserving statistical relationships.

6.3 Enhanced Conditional GAN

A GAN-based oversampling model built on **PyTorch**, designed to generate **label-conditioned** synthetic fraud samples with improved decision-boundary utility through classifier-aware loss.

- **Generator**: Combines noise and label embedding, processes through fully connected layers with BatchNorm + LeakyReLU to output synthetic samples.
- **Discriminator**: Takes real or synthetic samples with label embedding, outputs a critic score via MLP with BatchNorm + LeakyReLU.
- **Key Features**: Feature Matching Loss, Gradient Penalty (λ=10), EMA weight updates.
- **Training**: batch=256, epochs=500, Adam(lr=0.0002, β1=0.5, β2=0.999).

Algorithm 1: Enhanced Conditional GAN Training Procedure

Input: Real samples x_{real}, noise z
Output: Trained Generator G

1. Initialize Generator (G), Discriminator (D), and Classifier (C); set up EMA for G.
2. Pretrain C on x_{real} for feature extraction.
3. For each epoch:
   a. Discriminator step (repeat n_{critic} times):
      - Generate $x_{fake} = G(z)$
      - Compute L_D = BCE loss (real + fake) + gradient penalty
      - Update D with gradient clipping
   b. Generator step:
      - Generate $x_{fake} = G(z)$
      - Compute adversarial loss L_{adv} from D
      - Compute feature matching loss L_{fm} from C
      - $L_G = L_{adv} + \lambda_{fm} L_{fm}$
      - Update G and apply EMA update
   c. Adjust learning rates, check early stopping
4. Return EMA-smoothed G

7 Visualization Configuration

7.1 PCA

- **Purpose:** Reduce to 2D for scatter plot visualization of real vs. synthetic data
- **Library:** sklearn.decomposition.PCA
- **Settings:** n_components=2
- **Applied to:** Original dataset and oversampled datasets for visual comparison

7.2 t-SNE

- **Purpose:** Nonlinear visualization for distribution
- **Library:** sklearn.manifold.TSNE
- **Main Settings:** n_components=2, perplexity=30
- **Applied to:** Original dataset and oversampled datasets to check cluster separability

```
pca = PCA(n_components=2, random_state=42)
pca_results = pca.fit_transform(combined_data)

fig, axes = plt.subplots(1, 2, figsize=(15, 6))

sc1 = axes[0].scatter(pca_results[:, 0], pca_results[:, 1],
                      c=combined_labels, cmap='viridis', alpha=0.6)
axes[0].set_title('PCA - Colored by Class')
plt.colorbar(sc1, ax=axes[0])

sc2 = axes[1].scatter(pca_results[:, 0], pca_results[:, 1],
                      c=data_sources, cmap='Accent', alpha=0.6)
axes[1].set_title('PCA - Colored by Source')
cbar2 = plt.colorbar(sc2, ax=axes[1], ticks=[0, 1])
cbar2.ax.set_yticklabels(['Original', 'Synthetic'])

fig.suptitle(title)
plt.tight_layout()
plt.show()
```

```
tsne = TSNE(n_components=2, perplexity=perplexity, random_state=random_state)
tsne_results = tsne.fit_transform(combined_data)

fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sc1 = axes[0].scatter(tsne_results[:, 0], tsne_results[:, 1],
                      c=combined_labels, cmap='viridis', alpha=0.6)
axes[0].set_title('All Data (Colored by Class)')
plt.colorbar(sc1, ax=axes[0])

sc2 = axes[1].scatter(tsne_results[:, 0], tsne_results[:, 1],
                      c=data_sources, cmap='Accent', alpha=0.6)
axes[1].set_title('All Data (Colored by Source)')
cbar2 = plt.colorbar(sc2, ax=axes[1], ticks=[0, 1])
cbar2.ax.set_yticklabels(['Original', 'Synthetic'])

fraud_mask = combined_labels == 1
sc3 = axes[2].scatter(tsne_results[fraud_mask, 0], tsne_results[fraud_mask, 1],
                      c=data_sources[fraud_mask], cmap='Accent', alpha=0.7)
axes[2].set_title('Fraud Cases Only')
cbar3 = plt.colorbar(sc3, ax=axes[2], ticks=[0, 1])
cbar3.ax.set_yticklabels(['Original', 'Synthetic'])

fig.suptitle(title)
plt.tight_layout()
plt.show()
```

Figure 3: PCA and t-SNE Calculation

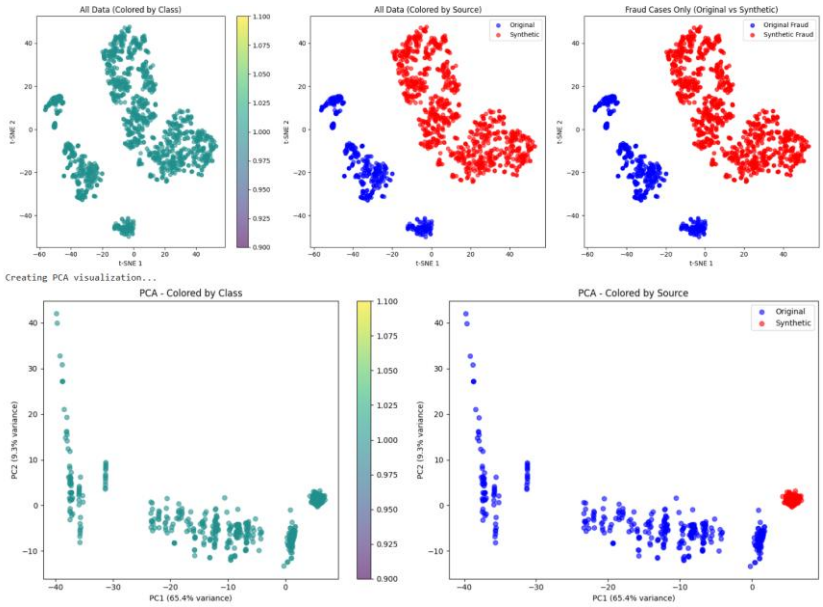

Figure 4: PCA and t-SNE of Real vs. Synthetic Fraud Data(Output)

8 Evaluation Methods

Table 3: Evaluation Methods

Criterion	Method	Notes
Precision / Recall / F1	sklearn.metrics	Important for class imbalance
ROC-AUC / PR-AUC	sklearn.metrics	PR-AUC is advantageous for rare classes
Specificity	sklearn.metrics	Ability to identify legitimate transactions
KS-Test	scipy.stats	Feature-wise distribution comparison
Jensen–Shannon Distance	scipy.spatial	Overall distribution similarity
Correlation Similarity	Pearson Corr.	Preservation of inter-feature structure

```python
def evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]
    precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, average='binary')
    roc_auc = roc_auc_score(y_test, y_prob)
    cm = confusion_matrix(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    rc, pr, _ = precision_recall_curve(y_test, y_prob)
    return {
        "precision": precision,
        "recall": recall,
        "f1": f1,
        "roc_auc": roc_auc,
        "confusion_matrix": cm,
        "roc_curve": (fpr, tpr),
        "pr_curve": (rc, pr)
    }
```

Figure 5: Evaluate Model

9 Sample Results Summary

Table 4: Classification Summary (Traditional Oversamplers & Enhanced GAN)

Method	Precision	Recall	F1-Score	ROC-AUC	PR-AUC	Specificity
No Oversampling	0.9412	0.8163	0.8743	0.9698	0.8616	0.9999
Enhanced GAN	0.9398	0.7959	0.8619	0.9620	0.8591	0.9999
RandomOverSampler	0.6457	0.8367	0.7289	0.9825	0.8265	0.9992
SMOTE	0.3818	0.8571	0.5283	0.9824	0.7897	0.9976
SMOTE-Tomek	0.3818	0.8571	0.5283	0.9824	0.7897	0.9976
ADASYN	0.1117	0.9082	0.1989	0.9831	0.7098	0.9875

Table 5: Classification Summary

Method	Precision	Recall	F1-Score	ROC-AUC	PR-AUC	Specificity
No Oversampling	0.9412	0.8163	0.8743	0.9698	0.8616	0.9999
Enhanced GAN	0.9398	0.7959	0.8619	0.9620	0.8591	0.9999
CTGAN	0.9213	0.8367	0.8770	0.9758	0.8532	0.9999
TVAE	0.8750	0.8041	0.8380	0.9752	0.8244	0.9998
Gaussian Copula	0.9187	0.7635	0.8339	0.9741	0.8090	0.9999

Key Insights:

- Enhanced GAN maintains a good balance between Precision and Recall, with an F1-score indicating stable performance.
- Traditional oversamplers generally increase recall at the cost of precision, whereas generative models vary in balance depending on architecture.
- Some generative models lack sufficient support for the decision boundary.

10 Visual and Statistical Analysis

- **t-SNE/PCA Visualization:** Enhanced GAN samples are close to real fraud data but not identical, providing useful diversity.
- **Correlation Analysis:** Some structural distortions exist, but performance is maintained.
- **KS/JSD Test:** Distribution differences are present but contribute to performance improvement.

Table 6: Statistical Fidelity Metrics

Metric	Result	Interpretation
Kolmogorov–Smirnov (KS) Test	$p \leq 0.05$ for all 30 features	Significant distribution differences between real and synthetic data
Jensen–Shannon Distance (JSD)	0.5750	Moderate level of distribution difference
Correlation Structure Similarity	–0.2463	Some distortion in inter-feature correlation structure

References

UCI Machine Learning (2016). *Default of Credit Card Clients Dataset*. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud [Accessed 9 Aug. 2025].

Pandas (2024). *pandas documentation — pandas 1.0.1 documentation*. [online] pandas.pydata.org. Available at: https://pandas.pydata.org/docs/.

Scikit-learn.org. (2019). *Documentation scikit-learn: machine learning in Python — scikit-learn 0.21.2 documentation*. [online] Available at: https://scikit-learn.org/stable/documentation.html.

Imbalanced-learn.org. (2021). *imbalanced-learn documentation — Version 0.8.1*. [online] Available at: https://imbalanced-learn.org/stable/.

PyTorch (2019). *PyTorch documentation — PyTorch master documentation*. [online] Pytorch.org. Available at: https://pytorch.org/docs/stable/index.html.

Sdv.dev. (2025). *Welcome to the SDV! | Synthetic Data Vault*. [online] Available at: https://docs.sdv.dev/ [Accessed 9 Aug. 2025].