Channel Islands

CALIFORNIA STATE UNIVERSITY

StatGenerative Model

A Thesis Presented to

The Faculty of the Computer Science Department

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

by

Student Name:
Abdul Quadir Owais, Muhammed

Advisor:
Dr.Micheal Soltys

Month Year

Dec 2025

APPROVED FOR MS IN COMPUTER SCIENCE

Dr. Micheal Soltys 9 DEC 2025

Advisor: Advisor name **Date**

Dr. Jason Isaacs 9 DEC 2025

Name **Date**

Dr. Scott Feister 9 DEC 2025

Name **Date**

APPROVED FOR THE UNIVERSITY

Abdul Quadir Owais, Muhammed 9 DEC 2025

Name **Date**

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

StatGenerative Model

Title of Item

Synthetic Data Generation, CTGAN, AKDE

3 to 5 keywords or phrases to describe the item

Abdul Quadir Owais, Muhammed

Author(s) Name (Print)

9 DEC 2025

Author(s) Signature **Date**

Contents

1. Introduction

1.1 Background & Motivation

In recent years, the proliferation of machine learning (ML) applications across domains such as healthcare, finance, and social sciences has significantly increased the demand for high-quality datasets [1, 2]. Synthetic data refers to artificially generated records that mimic the statistical properties of real-world data without directly exposing sensitive information [3].
This approach offers several motivations:

Augmenting Small Datasets

In many applications, especially in academic or resource-constrained environments, the available dataset size is insufficient to train robust machine learning models. Synthetic data can augment small datasets by providing additional samples that follow similar distributions, thereby improving generalization and reducing overfitting [2, 6].

Robustness and Bias Mitigation

Real-world datasets often suffer from class imbalance, missing data, or inherent biases. Synthetic data generation allows controlled augmentation of underrepresented groups or scenarios, leading to fairer and more robust models [7, 8].

Cost-Effective Data Availability

Collecting and annotating large datasets is labor-intensive and expensive. Synthetic data offers a scalable alternative by enabling automated generation of realistic data points at significantly lower cost [9].

Applications Across Domains

Healthcare: creating synthetic patient records for predictive modeling without exposing personal health data [4, 5].

Finance: simulating customer transactions or fraud detection scenarios for algorithm testing [10].

Autonomous Driving: generating diverse driving conditions, weather scenarios, and rare events for training perception models [11].

Cybersecurity: synthesizing network traffic data to improve intrusion detection systems [12].

Given these motivations, research into synthetic data generation has gained momentum, particularly with the rise of generative models such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and nonparametric methods like Kernel Density Estimation (KDE) [13, 14, 15].

This thesis builds upon these motivations by exploring and comparing different synthetic data generation methods—including KDE-based approaches and GAN-based models—on benchmark datasets. The ultimate goal is to evaluate their effectiveness, highlight their limitations, and provide practical insights into their potential use cases.

1.2 Problem Statement

Despite significant advancements in machine learning and generative modeling, the process of generating high-fidelity synthetic data remains a complex and unsolved challenge [1, 2]. The core problem lies in achieving a delicate balance between statistical similarity, explainability and utility [16, 17]. While synthetic data should closely approximate the real dataset's distribution, it must not reproduce or reveal any sensitive information about individuals or proprietary data sources [4].

Several practical and theoretical gaps motivate this research:

Distributional Fidelity vs. Generalization

Many existing generative models, such as CTGAN or VAEs, focus on replicating observed data patterns. However, they often fail to generalize beyond the training data distribution, resulting in synthetic samples that either overfit or exhibit unrealistic combinations of features. Further sometimes they behave absurdly on the certain datasets giving unrealistic generated data which has less to do with any similarity of data [13, 14, 18].

Lack of Adaptive Frameworks

Existing models typically rely on fixed assumptions about the data distribution. Methods like standard KDE assume a global bandwidth parameter, ignoring local variations in data density [15, 19]. Adaptive KDE, on the other hand, offers a dynamic approach but remains underexplored in practical synthetic data generation pipelines.

Given these challenges, this thesis aims to develop and compare a range of synthetic data generation techniques—spanning from parametric (GANs, VAEs) to

nonparametric (Adaptive KDE) models—applied to real-world datasets such as the Bean's dataset [20]. The comparison will be based on metrics for fidelity, diversity, and downstream model performance.

By systematically evaluating these methods, this research seeks to answer the following key questions:

1. How effectively can statistical (KDE-based) methods approximate real data distributions compared to neural network-based generative models?
2. What trade-offs exist between complexity, interpretability, and accuracy across different synthetic data generation techniques?
3. Can adaptive, kernel-based methods serve as lightweight yet effective alternatives to complex deep generative models?

This thesis ultimately aims to bridge the gap between statistical simplicity and generative power, providing a comparative analysis and practical insights that can guide future research and applications in synthetic data generation.

1.3 Objectives and Research Questions

Overview

The core aim of this thesis is to **investigate, visualize, and compare** the performance of two fundamentally different approaches to synthetic data generation—**CTGAN (Conditional Tabular GAN)** and **Adaptive Kernel Density Estimation (AKDE)**—on structured tabular data, using the **Beans dataset** as the main case study.

While quantitative metrics are important, this research places a **strong emphasis on visual interpretability**, enabling an intuitive and exploratory understanding of how each model reconstructs and deviates from real data distributions. Through histograms, correlation heatmaps, pairwise feature comparisons, and dimensionality reduction visualizations (PCA), the study aims to uncover nuanced differences in generative behavior that may not be captured by numerical metrics alone.

Primary Objectives

1. **To analyze and visualize the distributional fidelity** of synthetic data generated by CTGAN and AKDE, comparing them against the real Beans dataset.
2. **To design a reproducible framework** for evaluating synthetic data quality using both statistical and visual techniques—focusing on feature-wise density comparisons, correlation preservation, and class-wise separability.
3. **To identify strengths and weaknesses** of deep generative (CTGAN) versus nonparametric statistical (AKDE) methods in terms of:
    - Reproducing multi-modal and non-Gaussian distributions.
    - Maintaining inter-feature relationships and covariance structures.
    - Scalability and interpretability.
4. **To demonstrate the value of visualization-driven evaluation**, highlighting cases where visual diagnostics (e.g., KDE plots, scatter matrices) reveal discrepancies that may be overlooked by scalar metrics like MSE.
5. **To contribute a practical, open-source toolkit** (via the `SynExploration` repository) that has both CTGAN and AKDE pipelines, supporting future studies in tabular synthetic data generation.

Research Questions

1. **How visually and statistically similar** are the synthetic distributions produced by CTGAN and AKDE when compared with the real Beans dataset?

2. **Which approach better preserves inter-feature relationships and class separability**, as observed through pair plots, PCA projections, and correlation matrices?

3. **Does AKDE offer a computationally efficient yet accurate alternative** to deep generative methods like CTGAN for structured tabular data?

4. **Can visualization-based analysis complement or even surpass traditional statistical metrics** in assessing synthetic data quality?

1.4 Scope and Limitations

Scope

This thesis is confined to the **generation and visual evaluation of synthetic tabular data**, with a focus on **comparing CTGAN (Conditional Tabular GAN)** and **Adaptive Kernel Density Estimation (AKDE)** techniques using the **Beans dataset** [20].

The primary emphasis is placed on **visual exploration and interpretability**, rather than solely on numerical benchmarks. The study leverages visualization as a diagnostic and comparative tool, enabling a deeper understanding of how each model learns and reconstructs the underlying data distribution. [21, 22]

The scope of this research includes:

1. **Data Domain [20]**
    - Focused exclusively on **tabular data** (Beans dataset).
    - Each feature is numerical or categorical and suitable for both GAN-based and KDE-based modeling.
2. **Models Under Study [14, 15, 19]**
    - **CTGAN** – a deep learning-based generative model specialized for tabular data with mixed feature types and conditional sampling.
    - **AKDE** – a statistical, nonparametric model that adapts kernel bandwidths locally, providing smooth and interpretable distribution approximations.
3. **Visualization-Driven Analysis [21, 22]**
    - Distribution comparison plots (histograms, KDE overlays).
    - Dimensionality reduction (PCA) to compare structural similarity.
    - Correlation heatmaps to assess inter-feature consistency.
    - Pairwise scatter matrices to identify visual deviations between real and synthetic samples.
4. **Evaluation Metrics [8, 7, 15]**
    - Statistical metrics such as KS-test, Wasserstein Distance, and correlation difference.
    - Visual fidelity indicators for interpretability.
    - Qualitative assessment supported by plotted visual evidence.
5. **Implementation and Tools**
    - All experiments are implemented in Python using libraries such as PyTorch, CTGAN, NumPy, SciPy, and Matplotlib.
    - Reproducible pipeline hosted on the **SynExploration** GitHub repository.

Limitations

The scope of this thesis is intentionally constrained to the exploration and comparative evaluation of two synthetic data generation techniques—Adaptive Kernel Density Estimation (AKDE) and Conditional Tabular GAN (CTGAN)—applied to the *Dry Beans* dataset.

While the findings provide valuable insights into the behavior of statistical and deep generative models on structured tabular data, several limitations must be acknowledged:

- **Single-domain evaluation:** Results are limited to a single dataset (*Dry Beans*), which primarily contains agricultural morphological features. Consequently, the conclusions drawn may not generalize to other domains such as healthcare, finance, or genomics, where data complexity and feature dependencies differ substantially.

- **Limited sample size:** The sample size (n = 500) used for training may be insufficient for CTGAN's adversarial training to reach full convergence. Deep generative models typically benefit from larger data volumes to stabilize discriminator–generator dynamics and learn higher-order dependencies.

- **Absence of privacy analysis:** Although synthetic data generation is often motivated by privacy preservation, no formal privacy guarantees—such as *differential privacy*, *PATE-GAN frameworks*, or *membership inference resistance*—were implemented or evaluated in this work.
- **Evaluation scope:** This study focuses primarily on statistical fidelity, visualization, and downstream classification utility, without extensive exploration of scalability or hyperparameter robustness across multiple datasets.

Despite these constraints, the controlled setup enables a clear and reproducible analysis of core model characteristics and provides a strong foundation for future cross-domain studies.

1.5 Thesis Outline

This thesis is organized into six main chapters, each serving a distinct purpose in the overall research flow — from theoretical grounding to practical experimentation and evaluation.

Chapter 1 – Introduction

The opening chapter establishes the motivation behind synthetic data generation, the growing demand for augmentation-based data strategies, and introduces the two key generative frameworks under investigation: **CTGAN** and **Adaptive KDE**. It outlines the problem statement, objectives, research questions, scope, and limitations of the study, providing a foundation for the research that follows.

Chapter 2 – Literature Review

This chapter surveys existing research on **synthetic data generation techniques**, highlighting key developments in deep generative models (GANs, VAEs, CTGANs) and non-parametric statistical models (KDE, AKDE). It also reviews evaluation metrics, visualization methodologies, and prior comparative analyses, identifying gaps that this study addresses — particularly the lack of **visual interpretability-focused comparisons** between statistical and neural approaches for tabular data.

Chapter 3 – Methodology

Chapter 3 details the **Beans dataset** used as the primary benchmark, describing its features, preprocessing steps, and data characteristics. It then introduces the **experimental pipeline**, including:

- Implementation of **CTGAN** and **AKDE** models.
- Data preprocessing and feature normalization techniques.
- Synthetic data generation procedures.
- Visualization and evaluation frameworks (histograms, KDE plots, PCA, correlation heatmaps).
  The chapter concludes with the computational setup and reproducibility strategy linking to the **SynExploration GitHub repository**.

Chapter 4 – Experiments and Results

This chapter presents and compares the outcomes of both models across multiple visual and quantitative perspectives. It includes:

- Univariate and multivariate distribution visualizations.
- Feature-wise density plots and pairwise scatter comparisons.
- Dimensionality-reduction views (PCA) for class separation.
- Statistical metrics such as KS-statistic, correlation difference, and Wasserstein Distance.

  Each section interprets the visual patterns to assess the realism, diversity, and fidelity of the generated data relative to the real Beans dataset.

Chapter 5 – Conclusion and Future Work

The concluding chapter summarizes the major findings, contributions, and implications of the study. It also identifies potential directions for future research, including the integration of hybrid generative models, improved visual evaluation metrics, and expansion to larger or more complex datasets.

2. Literature Review

2.1 Synthetic Data Generation: Overview

Synthetic data generation refers to the process of creating artificial data that preserves the statistical characteristics and relationships of real-world datasets while avoiding direct disclosure of sensitive or proprietary information [1, 2, 3].

2.1.1 Generative Model Paradigms

The evolution of synthetic data techniques can be divided into three broad paradigms:

(a) Statistical and Rule-Based Approaches

Early methods relied on parametric models, random sampling, or bootstrapping. While computationally inexpensive and interpretable, they fail to capture complex non-linear relationships in multi-modal datasets [15].
Kernel Density Estimation (KDE) improved upon these techniques by providing nonparametric density estimation, and its adaptive variant (AKDE) further refined local bandwidth control for variable-density regions [19, 17].

(b) Deep Generative Models

The advent of **Generative Adversarial Networks (GANs)** [13] revolutionized synthetic data generation. GANs consist of a generator–discriminator pair trained in adversarial fashion, producing highly realistic data samples.
Subsequent architectures specialized this framework for domain-specific use cases—e.g., **CTGAN** [14] for *tabular* data, **MedGAN** [18] for healthcare records, and **TimeGAN** for temporal series.

GANs effectively learn high-dimensional joint distributions but suffer from *mode collapse*, instability, and poor interpretability [13, 14].

2.1.2 Comparative View: Statistical vs. Neural Models

Aspect	CTGAN (Deep Generative)	Adaptive KDE (Statistical)
Model Type	Deep neural network (GAN)	Nonparametric density estimator
Data Suitability	Mixed tabular data with categorical features	Continuous tabular data

Explainability	Limited (black-box)	High (interpretable kernels and bandwidths)
Computation	GPU intensive training	Lightweight, CPU-friendly
Fidelity	Learns complex joint dependencies	Captures local density variations
Weaknesses	Prone to mode collapse and instability	Sensitive to bandwidth selection

This contrast underscores the complementary nature of CTGAN and AKDE: CTGAN provides flexibility and generative depth, whereas AKDE offers statistical rigor and interpretability.

By studying both under a unified visual-analytic framework, this thesis explores how statistical transparency can complement neural realism.

2.1.3 Evaluation Challenges

Assessing synthetic data quality is inherently multifaceted [8]. Common quantitative metrics include the **Kolmogorov–Smirnov (KS) test**, **Wasserstein Distance**, and **correlation difference**, which measure feature-wise or distributional similarity [15].

However, numerical scores alone may overlook structural or visual discrepancies. Therefore, **visual evaluation**—through histograms, kernel overlays, pair plots, correlation maps, PCA projections—has emerged as a critical complementary approach [21, 22].

2.1.4 Summary

The literature demonstrates strong progress in both deep and statistical synthetic-data modeling, yet there remains a gap in **comparative, visualization-driven analyses** of these two paradigms.

Most prior works prioritize quantitative fidelity metrics, leaving interpretability and visual understanding under-explored—particularly for tabular datasets such as **Beans**.

This research aims to fill that gap by developing a visual-analytic framework to evaluate **CTGAN** and **AKDE** side by side.

2.2 Existing Methods and Approaches

The domain of synthetic data generation encompasses a diverse set of methodologies that range from statistical density estimation to deep generative architectures. Each approach presents unique trade-offs in terms of fidelity, interpretability, and computational cost preservation. This section reviews the most prominent techniques and comparative studies relevant to **tabular data synthesis**, with special attention to models such as **CTGAN**, **Variational Autoencoders (VAEs)**, and **Adaptive Kernel Density Estimation (AKDE)**.

2.2.1 Generative Adversarial Networks (GANs)

Since their introduction by **Goodfellow et al. (2014)** [**13**], GANs have been a central paradigm for data synthesis. A GAN comprises a *generator* that learns to produce realistic samples from random noise and a *discriminator* that differentiates between real and synthetic data. Training progresses as a two-player minimax game until the generator's distribution p_g(x) approximates the real data distribution p_data(x).

While the original formulation targeted image data, subsequent research adapted GANs for **tabular and structured data**, resulting in variants such as **CTGAN** [14], **MedGAN** [18], **TableGAN**, and **CopulaGAN**.
These models address the discrete-continuous feature mix through conditional sampling and embedding mechanisms.

- **CTGAN** [14] uses a *conditional vector sampling strategy* to overcome class imbalance and mode collapse.
- **TableGAN** employs convolutional filters to exploit spatial correlations between tabular features.
- **CopulaGAN** integrates copula transformations before adversarial training to stabilize feature dependencies.

Despite their expressive power, GAN-based models suffer from several drawbacks: (1) unstable convergence; (2) sensitivity to hyperparameters; (3) difficulty in reproducing exact statistical properties; and (4) limited interpretability due to their black-box architecture [13, 14].

2.2.2 Variational Autoencoders (VAEs)

Variational Autoencoders, proposed by **Kingma & Welling (2014)** [16], form another deep generative family that models data through latent-variable distributions. Unlike GANs, VAEs optimize a reconstruction loss and a regularization term (KL-divergence) to ensure the latent space follows a known prior (typically Gaussian).

Recent adaptations such as **TVAE (Tabular VAE)** and **CT-VAE** aim to synthesize structured tabular data [6]. VAEs are comparatively easier to train and more stable than GANs, though they often generate blurrier or less precise samples because of their continuous latent approximations.

For tabular data, VAEs require careful encoding of categorical variables and are often combined with mixture-of-Gaussians priors to capture multimodal behavior. However, their generated data may lack fine-grained feature correlations observable in models like CTGAN [14, 18].

2.2.3 Kernel Density Estimation (KDE) and Adaptive KDE (AKDE)

In contrast to deep models, **Kernel Density Estimation (KDE)** [15] provides a **nonparametric statistical approach** to approximate the underlying probability density function (PDF) of a dataset. Given nnn samples x1,x2,...,xnx_1, x_2, ..., x_nx1,x2,...,xn, the KDE estimator is defined as:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where K is a kernel function (commonly Gaussian) and h is the bandwidth controlling the smoothness of the estimated density.

However, **standard KDE** assumes a *global* bandwidth h, which can over-smooth dense regions and under-smooth sparse ones. To overcome this, **Adaptive Kernel Density Estimation (AKDE)** [**19**, **17**] introduces *locally varying bandwidths* hih_ihi proportional to the inverse of the local data density, following the rule:

$$h_i = h \left[\frac{\hat{f}(x_i)}{g}\right]^{-1/2}$$

where g is the geometric mean of all **f(xi)**.
AKDE thereby adapts the kernel width dynamically—resulting in better representation of multimodal and heteroscedastic data, as often found in the **Beans dataset**.

Compared with CTGAN, AKDE is **computationally lightweight**, **transparent**, and **interpretable**, but it struggles with **high-dimensional scaling** and **categorical variables**. Despite these constraints, AKDE remains an appealing baseline for

explainable synthetic data generation, particularly when the goal is statistical realism rather than visual indistinguishability [15, 19].

2.2.4 Comparative Evaluation Frameworks

The performance of synthetic data generators has been evaluated through both **quantitative** and **qualitative** approaches:

1. **Statistical Metrics:**
    - **Kolmogorov–Smirnov (KS) test**, **Wasserstein Distance**, **Jensen–Shannon Divergence**, and **Correlation Preservation** are commonly used to assess distributional similarity [8, 15].

2. **Task-Driven Metrics:**
    - Performance of downstream ML models (e.g., classifier trained on synthetic data and tested on real data) is used as an indirect fidelity indicator [7, 8].

3. **Visual Evaluation:**
    - Visualization techniques—such as pairwise scatter plots, histograms, PCA projections—help qualitatively identify differences between real and generated distributions [21, 22].
    - Such visual analyses are especially valuable for interpretability, as they reveal structural patterns (e.g., clustering or feature correlation) often masked by scalar metrics.

2.2.5 Summary of Existing Approaches

Method	Category	Key Advantage	Primary Limitation	Typical Use Case
CTGAN	Deep Generative (GAN)	Handles mixed data; captures nonlinear dependencies	Training instability; low interpretability	Tabular datasets with mixed types
VAE / TVAE	Deep Generative (Autoencoder)	Stable and structured latent space	Produces smoother, less precise samples	Statistical or semi-supervised tabular synthesis

KDE	Statistical Nonparametric	Simple and interpretable	Over/under-smoothing; global bandwidth limitation	Continuous low-dimensional data
AKDE	Adaptive Nonparametric	Local density adaptation; interpretable	Not scalable to high-D data; less suited for categorical features	Explainable synthetic tabular data

---

Collectively, these studies illustrate that **deep generative models** (e.g., CTGAN) achieve high fidelity at the expense of interpretability and compute, while **statistical approaches** (e.g., AKDE) offer transparency and stability but limited scalability. The coexistence of these paradigms motivates a hybrid research direction—one that this thesis explores through **visualization-driven comparison of CTGAN and AKDE on the Beans dataset**.

2.3 Gaps in the Literature

Although substantial progress has been made in the development of generative models and density-estimation techniques, several important gaps remain in the current body of research on **synthetic data generation for tabular datasets**.

2.3.1 Limited Comparative Analysis Between Statistical and Neural Models

Most prior studies have focused on the **performance of a single model family**, such as GAN-based frameworks or statistical estimators, without offering systematic comparisons between them.

For example, deep models like CTGAN [14] demonstrate high distributional fidelity but are often treated as black-box generators, while nonparametric techniques such as AKDE [15, 19] offer interpretability but are rarely evaluated side by side with neural counterparts.

This lack of comparative work leaves open questions about the trade-offs between **explainability**, **fidelity**, and **computational complexity**—particularly in structured tabular domains such as the **Beans dataset** [20].

2.3.2 Under-representation of Visualization-Driven Evaluation

Synthetic data quality is still largely assessed using numerical metrics such as KS-statistic, Wasserstein Distance, or downstream ML accuracy [8, 15].
While these quantitative indicators are useful, they often fail to reveal deeper **structural and correlation-based discrepancies** between real and synthetic data.
Only a handful of works, such as Kim et al. (2023) [7] and Sacha et al. (2018) [21], emphasize **visual analytics** as a diagnostic tool.
Visual comparisons—histograms, KDE overlays, pairwise scatter plots, PCA projections—offer a more intuitive understanding of data realism and are essential for explainable synthetic data generation.
The existing literature rarely incorporates such visualization frameworks into a reproducible experimental methodology.

2.3.3 Insufficient Exploration of Adaptive KDE for Synthetic Data

While KDE and its adaptive variant (AKDE) have been well studied for density estimation [15, 17, 19], their application as *stand-alone synthetic data generators* remains underexplored.
Few studies investigate how bandwidth adaptation affects the fidelity of generated samples or how AKDE performs relative to deep generative models when evaluated both statistically and visually.
Furthermore, most comparative papers focus on image or time-series data, leaving **tabular and mixed-type data** comparatively neglected.

2.3.4 Reproducibility and Transparency Challenges

Deep generative models, especially GAN-based ones, often exhibit **hyperparameter sensitivity** and non-deterministic training outcomes [13, 14, 18].
Small variations in random seeds, optimization strategies, or hardware configurations can lead to markedly different synthetic outputs, complicating cross-study comparisons.
Conversely, statistical models like AKDE are fully deterministic yet rarely benchmarked under standardized pipelines.
This lack of **transparent, reproducible experimentation** hinders the establishment of reliable baselines and fair evaluation of generative techniques [6, 8].

2.3.5 Neglect of Mid-Sized, Real-World Tabular Datasets

A majority of synthetic-data research either employs **toy datasets** (e.g., Gaussian mixtures) for simplicity or **large industrial datasets** for benchmarking [1, 2].

Mid-scale datasets such as the **Beans dataset** [**20**]—which feature moderate dimensionality and interpretable features—are ideal for combining **quantitative** and **visual** evaluation, yet are rarely utilized.
This oversight limits the generalizability and interpretability of prior results.

2.3.6 Summary of Identified Gaps

Gap Area	Key Limitation in Literature	Relevance to This Thesis
Comparative Frameworks	Few studies directly compare CTGAN (deep) vs AKDE (statistical)	Core contribution: side-by-side comparative analysis
Visualization Based Evaluation	Over-reliance on scalar metrics	Introduces systematic visual diagnostics
Adaptive KDE Utilization	Rarely used as generative model	Expands AKDE for data synthesis
Reproducibility	Inconsistent benchmarking across models	Provides open, reproducible GitHub pipeline
Dataset Choice	Overuse of toy/industrial extremes	Employs mid-scale, interpretable Beans dataset

2.3.7 Research Justification

The literature thus lacks a **visual-analytic, reproducible comparison** between interpretable statistical models and deep neural generators for synthetic tabular data.
This thesis directly addresses these deficiencies by:

1. Reproducible AKDE pipeline and a matched CTGAN baseline trained via SDV; both are evaluated under a common protocol
2. Introducing visualization-driven evaluation alongside statistical metrics.
3. Demonstrating results on the Beans dataset to illustrate interpretable, real-world relevance.

By filling these gaps, the study contributes to both the methodological rigor and explainable evaluation of synthetic data generation.

2.4 Summary

This chapter reviewed the theoretical and empirical foundations of **synthetic data generation**, encompassing both deep learning–based and statistical approaches. The discussion highlighted how the evolution of this field has shifted from classical **density estimation techniques** toward **adversarial and variational deep generative models**, motivated by the need for augmentation, and realistic simulation across domains such as healthcare, finance, and autonomous systems.

In **Section 2.1**, an overview of synthetic data generation established its growing significance in modern machine learning workflows. It outlined how methods such as GANs and VAEs have expanded the ability to model complex, nonlinear data distributions, while statistical approaches like KDE and AKDE provide transparent and interpretable density estimates.

Section 2.2 examined existing methodologies, categorizing them into **deep generative** and **nonparametric statistical** families. Deep learning models, including **CTGAN**, demonstrated the capacity to synthesize mixed-type tabular data with high fidelity but suffer from interpretability and stability issues. Conversely, **Adaptive Kernel Density Estimation (AKDE)** offers a lightweight and interpretable framework but has been underutilized in the context of synthetic data generation. The section also summarized common evaluation techniques, distinguishing between **quantitative metrics** (e.g., KS test, Wasserstein Distance) and **qualitative visualization-based assessments** (e.g., PCA, and histogram overlays).

In **Section 2.3**, several gaps in the literature were identified:

1. The absence of direct **comparative studies** between deep generative and statistical density-based models for tabular data.
2. A lack of **visualization-driven evaluation frameworks** that combine statistical analysis with visual interpretability.
3. Minimal exploration of **adaptive KDE** as a standalone synthetic data generator.
4. Ongoing challenges in **reproducibility and transparency**, particularly for GAN-based methods.
5. Limited focus on **mid-scale tabular datasets**, such as the Beans dataset, which provide both interpretability and practical complexity.

These identified gaps form the **foundation and motivation** for this thesis. By addressing them, the research contributes a **reproducible, visual-analytic framework** that unites **CTGAN** (deep generative) and **AKDE** (statistical) paradigms under a single comparative study.

The next chapter, **Methodology (Chapter 3)**, presents the experimental design, datasets, preprocessing steps, implementation details, and the visualization-based evaluation framework developed to conduct this comparative analysis.

---

3. Methodology

3.1 Dataset Description

3.1.1 Data Source

The dataset used in this study is the **Dry Beans Dataset**, obtained from the **UCI Machine Learning Repository** [**20**].
This dataset was originally developed for the classification of **dry bean seed varieties** using visual and morphological characteristics derived from digital images of the seeds. It provides a well-balanced and moderately complex example of tabular data suitable for evaluating synthetic data generation models such as **CTGAN** and **Adaptive Kernel Density Estimation (AKDE)**.

The dataset contains **13,611 samples** and **17 attributes** (16 features and 1 class label). Each record represents a single bean seed characterized by shape, size, and geometric features extracted from an image.
The features are both **continuous** and **categorical**, making it ideal for mixed-type modeling.

The features include:

Feature Name	Description	Type
Area	Area of the bean (number of pixels within the boundary)	Continuous
Perimeter	Perimeter length of the bean outline	Continuous
MajorAxisLength	Length of the major axis of the ellipse that has the same normalized second central moments as the bean region	Continuous

MinorAxisLength	Length of the minor axis of the ellipse that has the same normalized second central moments as the bean region	Continuous
AspectRatio	Ratio of MajorAxisLength to MinorAxisLength	Continuous
Eccentricity	Measure of how much the bean shape deviates from a circle	Continuous
ConvexArea	Number of pixels in the convex hull of the bean region	Continuous
EquivDiameter	Diameter of a circle with the same area as the bean region	Continuous
Extent	Ratio of the area of the bean region to the bounding box area	Continuous
Solidity	Ratio of area to convex area	Continuous
Roundness	Measure of circularity of the bean	Continuous
Compactness	Measure of shape compactness	Continuous
ShapeFactor1	Shape descriptor based on boundary complexity	Continuous
ShapeFactor2	Shape descriptor variant for convexity	Continuous
ShapeFactor3	Ratio of perimeter² to area	Continuous
ShapeFactor 4	Ratio of area to major axis length²	Continuous
Class	Bean variety (e.g., Seker, Barbunya, Bombay, Cali, Dermosan, Horoz, Sira)	Categorical

3.1.2 Dataset Characteristics

- **Total Samples:** 13,611
- **Features:** 16 numerical features + 1 categorical class feature
- **Classes:** 7 (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz, Sira)
- **Data Type:** Tabular (continuous + categorical)
- **Source Format:** CSV file (downloaded from UCI repository)

This dataset is especially appropriate for **synthetic data generation experiments** because it represents a **mid-scale, interpretable dataset**—neither too small to limit model learning nor too large to cause computational inefficiency.
Among the seven available bean types, this study focuses on **five representative classes**—**SEKER**, **BARBUNYA**, **BOMBAY**, **CALI**, and **HOROZ**—selected to maintain both diversity and manageable computational scope.

3.1.3 Sampling and Experimental Subset

For the purpose of controlled experimentation and visualization, a subset of **500 samples** was randomly selected from the full dataset.
This decision was made to:

1. Maintain a balanced representation across the chosen bean classes,
2. Ensure computational feasibility when training and evaluating multiple generative models (CTGAN and AKDE), and
3. Facilitate clear visualization of the generated distributions in two- and three-dimensional feature spaces.

This reduced sample size also allows for rapid iteration when evaluating visualization-driven metrics such as **kernel overlays**, **scatter matrices**, and **PCA projections**.

3.1.4 Preprocessing and Feature Engineering

To ensure compatibility between **CTGAN** and **AKDE**, and to maintain consistency across experiments, the dataset underwent several preprocessing steps:

1. **Data Cleaning**
    - No missing values were found in the dataset.
2. **Encoding of Categorical Variables**
    - The target variable, *Class*, was **label-encoded** to integer values (0–6) for modeling convenience.

- For CTGAN, categorical features were one-hot encoded internally during model training.

3. **Normalization of Continuous Variables**
   - Unlike earlier iterations, this version **does apply normalization** to stabilize kernel bandwidths and distance computations:
     i. A **reversible scaler** is fit **per class** (default: `StandardScaler`; optional `MinMaxScaler`), transforming the class's numeric features into a normalized space.
     ii. **AKDE is fit and sampled in normalized space**, then all synthetic samples are **inverse-transformed back** to the **original units** before saving.
   - Rationale: class-conditional normalization makes **k-NN distance–based bandwidths** comparable across features, reduces dominance by high-variance columns, and **improves stability** of adaptive KDE without sacrificing interpretability (thanks to the inverse transform).
4. **Outlier Detection and Handling**
   - Similarly, **no outlier detection or removal** was performed.
     This decision was motivated by the goal of evaluating whether synthetic data models can **replicate distributional irregularities**, including rare or extreme data points, rather than merely learning the central trends.
     The presence of outliers is particularly important for assessing how statistical estimators like **AKDE** adjust local bandwidths in low-density regions compared to the global distributional modeling of **CTGAN**.
5. **Train-Test Split (for Evaluation)**
   - In this experiment, the dataset was used **as a complete unit without a train–test split**.
   - Since the objective of this study is **data generation** and **visual comparison**, not predictive modeling, the entire subset of 500 samples was utilized to train each generative model.
   - The models then produced synthetic samples with matching dimensionality and sample size (i.e., *n_samples = 500*) for one-to-one visual and statistical comparison.

3.1.5 Summary

In summary, this study employs a **500-sample subset of the Dry Beans dataset**, focusing on **five bean varieties** and retaining **raw, unnormalized features** to examine the capacity of CTGAN and AKDE to model and regenerate unprocessed real-world distributions.

By intentionally preserving outliers and feature-scale diversity, the dataset serves as a

robust benchmark for **visual, statistical, and interpretability-based evaluation** of synthetic data generation models.

3.2 Data Preprocessing and Feature Handling

3.2.1 Overview

As described in Section 3.1, the selected subset of the **Dry Beans dataset** consisted of 500 unnormalized samples across five bean varieties. Minimal preprocessing was intentionally employed to evaluate whether the generative models—**CTGAN** and **Adaptive KDE (AKDE)**—can inherently learn the natural scale, variance, and outlier structure of the raw data without human-induced transformations.

This section focuses on how each model interprets and processes the raw tabular features differently.

3.2.2 Feature Encoding and Model Compatibility

CTGAN:
CTGAN supports both continuous and categorical data through its internal preprocessing pipeline. Categorical features are one-hot encoded and incorporated into a *conditional vector* that controls the generator's sampling distribution. Continuous features are transformed using **variational Gaussian mixture normalization**, which models non-Gaussian and multimodal distributions before rescaling them during generation. This mechanism allows CTGAN to handle raw data effectively, even without external normalization.

AKDE:
AKDE operates entirely in the continuous domain and directly estimates a **probability density function** for each feature using Gaussian kernels. Unlike CTGAN, AKDE does not require feature encoding or normalization—it adapts bandwidth locally based on the data's intrinsic density. This allows it to capture both smooth and irregular regions of the distribution, including outliers, by widening or narrowing its local kernels accordingly.

3.2.3 Comparative Preprocessing Requirements

Step	CTGAN	AKDE
Input Type	Mixed (numerical + categorical)	Continuous only
Feature Scaling	Internally normalized via Gaussian mixtures	None required
Outlier Treatment	Learns as part of data distribution	Captured via bandwidth adaptation
Encoding	One-hot for categorical features	Not applicable
Dependencies	PyTorch (via SDV/CTGAN)	NumPy, SciPy

Both models were trained using identical 500-sample inputs to ensure fair comparison. The absence of train–test splitting was consistent across both methods, as the objective was **distribution reconstruction** rather than predictive performance.

3.2.4 Implementation Note

The preprocessing workflow and data handling pipeline are implemented in Python using `pandas`, `numpy`, `scipy`, and `ctgan` libraries.

3.2.5 Summary

This section outlined how **CTGAN** and **AKDE** process tabular data under minimal preprocessing conditions.
While CTGAN leverages conditional encoding and internal normalization to learn complex feature dependencies, AKDE operates directly on raw continuous data, adjusting local densities through adaptive bandwidth selection.
These complementary preprocessing strategies establish the foundation for the comparative analysis presented in the following sections.

3.3 Model Implementation

3.3.1 Overview

This section describes the implementation of the two generative models used in this study:

1. **Conditional Tabular GAN (CTGAN)** – a deep generative neural model designed for mixed-type tabular data, and
2. **Adaptive Kernel Density Estimation (AKDE)** – a non-parametric statistical model that learns local probability densities via adaptive bandwidths.

Both models were implemented and executed in **Python**, with full reproducibility available in the public repository:
SynExploration GitHub Repository.

3.3.2 Conditional Tabular GAN (CTGAN)

Model Architecture

The **CTGAN** architecture consists of two multilayer perceptrons (MLPs):

- **Generator (G):**
  - Input: concatenation of a noise vector z (~ N(0, 1)) and a conditional vector c (one-hot encoding of the sampled categorical feature).
  - Layers: Fully connected dense layers (typically 3–4) with **LeakyReLU** activations and **Batch Normalization** for stability.
  - Output: synthetic tabular feature vector $\hat{x}$ matching the dimensionality of the real data.
- **Discriminator (D):**
  - Input: either a real record x or a generated record $\hat{x}$.
  - Layers: Dense MLPs with **LeakyReLU** activation, **Dropout**, and a final **sigmoid** output neuron.
  - Objective: maximize $\log D(x) + \log(1 - D(G(z, c)))$.

Training follows the **Wasserstein GAN with Gradient Penalty (WGAN-GP)** formulation to mitigate mode collapse and stabilize learning.

Training Pipeline

1. All 500 records from the Beans dataset (five features) were fed into the CTGAN model without normalization (as justified in Section 3.1).
2. During training, the model alternately updates G and D for each batch, conditioning on one categorical variable per iteration.
3. The generator learns to approximate the real data distribution *p_data(x)* by minimizing the Wasserstein distance between *p_g(x)* and *p_data(x)*.
4. Once trained, synthetic samples are generated by sampling random noise z and feeding it through G to produce 500 synthetic observations.

Implementation Details

Parameter	Value
Framework	SDV (Tabular) – CTGAN
Latent dimension	128
Batch size	500
Epochs	300
Learning rate	2×10^{-4}
Optimizer	Adam ($\beta_1 = 0.5$, $\beta_2 = 0.9$)
Conditional sampling	Enabled
Output size	500 samples

The CTGAN model was implemented using the `sdv.tabular.CTGAN` module in PyTorch.
Visual diagnostics—histograms, KDE overlays, PCA projections—were subsequently generated to evaluate distributional similarity between real and synthetic data.

3.3.3 Adaptive Kernel Density Estimation (AKDE)

Algorithmic Principle

The AKDE approach extends traditional KDE by dynamically adjusting the kernel bandwidth h_i for each sample x_i based on its local data density.
Given n samples $x_1,\ldots,x_n$, the adaptive bandwidths are computed using k-nearest-neighbor distances:

Step 1 – Compute Pairwise Distances:
For each x_i, compute Euclidean distances to all other points.

Step 2 – Local Bandwidth Estimation:
Determine $h_i = h \times$ mean(distance to k nearest neighbors).

Step 3 – Model Fitting:
For each x_i, fit a Gaussian kernel with bandwidth h_i using `sklearn.neighbors.KernelDensity`.

Step 4 – Synthetic Sampling:
Randomly select a fitted KDE model and draw one sample from its estimated distribution.

This process yields synthetic data that preserve both dense-region fidelity and sparse-region variability.

Implementation Structure

The implementation, derived from [test_akde-bean.ipynb], consists of modular Python functions:

- `load_data(file_path)`: Loads numeric columns from CSV input.
- `fit_adaptive_kde(data, bandwidth, k)`: Computes adaptive bandwidths via k-NN and fits Gaussian KDEs for each sample.
- `generate_adaptive_synthetic_data(kde_models, n_samples)`: Draws new samples from randomly chosen fitted KDEs.
- `save_synthetic_data(original_df, synthetic_data, output_file, Class)`: Saves generated synthetic samples as CSV.
- `main(file_path, output_file, Class, bandwidth=0.5, k=5, n_samples=500)`: Executes the full AKDE pipeline end-to-end.

Parameterization

Parameter	Value / Description
Kernel type	Gaussian
Initial bandwidth (h)	0.5

Adaptive neighbors (k)	5–10
Synthetic samples (n_samples)	500
Libraries used	scikit-learn, NumPy, SciPy, pandas

The adaptive bandwidth adjustment enables the model to produce realistic samples even in regions of varying density—effectively capturing both common and rare patterns observed in the real data.
Unlike CTGAN, which relies on high-dimensional latent representations, AKDE operates directly on empirical feature values, offering full interpretability of the synthetic generation process.

3.3.4 Hyperparameter justification

CTGAN Hyperparameters

Hyperparameters used:

- Latent dimension (`latent_dim`): 128
- Learning rate (`lr`): 2×10^{-4}
- Batch size: 500
- Epochs: 300
- Discriminator steps per generator step: 5
- Conditional sampling: Enabled

Justification:

> The CTGAN model was configured with parameters that balance training stability and model expressiveness, following standard guidelines from the original CTGAN paper (Xu et al., 2019) and subsequent implementations in the *Synthetic Data Vault (SDV)* library.
>
> The **latent dimension of 128** provides sufficient representational capacity to capture feature dependencies without overfitting small tabular datasets. A **learning rate of 2×10^{-4}**, tuned through limited experimentation (10^{-3}, 2×10^{-4}, 10^{-5}), achieved stable convergence without oscillation in generator loss.

A **batch size of 500** was chosen to ensure that each class in the reduced dataset was proportionally represented per training iteration, avoiding mode collapse. **300 epochs** were empirically sufficient for loss stabilization in both the generator and discriminator.

Conditional sampling was **enabled** to maintain class balance across the five bean types (`SEKER`, `BARBUNYA`, `BOMBAY`, `CALI`, `HOROZ`), ensuring each synthetic instance was generated relative to a specific class condition vector.

Although CTGAN can theoretically benefit from larger datasets, using **500 samples per class** allowed fair comparison with AKDE while controlling computational constraints on CPU-based experimentation.

AKDE Hyperparameters

Hyperparameters used:

- Base bandwidth (`h`): 0.5
- Number of neighbors (`k`): 5
- Kernel: Gaussian

Justification:

The Adaptive Kernel Density Estimation (AKDE) model extends classical KDE by adjusting local bandwidths based on neighborhood density. The **base bandwidth (h = 0.5)** was empirically selected after testing values {0.1, 0.3, 0.5, 1.0}. Smaller values produced noisy, overfitted density curves, while higher values excessively smoothed the distribution and blurred class boundaries.

The **number of neighbors (k = 5)** was selected to balance sensitivity to local variance with global smoothness. Increasing *k* beyond 10 produced diminishing improvements in the Wasserstein distance while increasing computational cost quadratically.

The **Gaussian kernel** was used as it provides a continuous and differentiable estimation suitable for continuous-valued features.

These parameters collectively allow AKDE to capture multi-modal feature densities and adapt its smoothing level to local data sparsity—especially in classes with fewer or more irregular samples. The choices align with prior literature where $k \in [5, 10]$ and $h \in [0.3, 1.0]$ provide optimal bias-variance trade-offs for small tabular datasets.

3.3.5 Comparative Perspective

Aspect	CTGAN	AKDE
Model Type	Deep Neural (GAN)	Statistical Non-parametric
Implementation Framework	PyTorch / SDV	scikit-learn / SciPy
Core Mechanism	Adversarial training (G–D pair)	Density estimation with adaptive bandwidth
Interpretability	Low (black-box)	High (statistical transparency)
Computational Load	High (GPU training)	Low (CPU feasible)
Data Sensitivity	Stable on normalized data	Robust to outliers and non-normalized data
Output Use	Complex tabular synthesis	Lightweight synthetic data augmentation

3.3.6 Summary

The **CTGAN** model provides a deep-learning-based, high-capacity generative framework capable of capturing complex non-linear dependencies in tabular data. In contrast, **AKDE** offers a transparent, mathematically grounded approach that emphasizes interpretability and local distribution preservation.

By employing both models within a unified experimental setup—drawing from the repository SynExploration—this study establishes a robust foundation for the **visual and statistical comparison** of synthetic data quality presented in the subsequent chapter.

3.4 Evaluation Metrics and Visualization Framework

3.4.1 Overview

Evaluating the quality of synthetic data is inherently multidimensional.
No single metric can fully capture the realism, diversity, and interpretability of generated samples.
Therefore, this study adopts a **hybrid evaluation framework** that combines both **quantitative statistical metrics** and **qualitative visualization techniques**.
The goal is to assess not only how closely synthetic data match real-data distributions but also how well the models preserve structural relationships and visual coherence across features.
This framework is inspired by prior comparative works on tabular synthetic data evaluation [**7**, **8**, **21**, **22**] and is fully reproducible through the visualization modules implemented in the public repository [23].

3.4.2 Statistical Significance of Evaluation Metrics

The fidelity of synthetic data was assessed using five complementary statistical metrics.
Each metric quantifies a distinct dimension of similarity between real and synthetic distributions.
The use of multiple independent indicators ensures multidimensional evaluation — distributional alignment, coverage, variance preservation, and inter-feature dependency.

1. Kolmogorov–Smirnov Statistic (KS_stat)

Purpose: Measures the **maximum divergence** between the empirical cumulative distribution functions (CDFs) of the real and synthetic data for each feature.

Mathematical Form:

$$D_{KS} = \sup_x |F_{real}(x) - F_{synthetic}(x)|$$

Interpretation:

- D_KS = 0 → identical distributions.
- Larger D_KS → greater discrepancy between real and synthetic cumulative distributions.

2. Wasserstein Distance (1st-Order Earth Mover's Distance)

Purpose: Quantifies the **minimum "effort"** required to transform one probability distribution into another, capturing both **shape** and **spread**.

Mathematical Form:

$$W_1(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \int |x - y| \, d\gamma(x, y)$$

Interpretation:

- Smaller values → distributions are more similar.
- Sensitive to outliers and overall mass displacement.

3. Coverage Percent

Purpose: Indicates the proportion of synthetic samples whose feature values **fall within the min–max range** of the corresponding real-data feature.

Mathematical Form:

$$\text{Coverage}(X) = \frac{1}{N} \sum_i \mathbb{I}(x_i^{syn} \in [\min(X_{real}), \max(X_{real})]) \times 100$$

Interpretation:

- High coverage → synthetic data respects the real feature bounds.
- Low coverage → over- or under-spreading of synthetic samples.

4. Variance Ratio

Purpose: Measures how well each model preserves the **relative feature variability** of the real dataset.

Mathematical Form:

$$VR = \frac{\mathrm{Var}(X_{syn})}{\mathrm{Var}(X_{real})}$$

Interpretation:

- VR ≈ 1: perfect variance preservation.
- VR > 1: synthetic data overly dispersed.
- VR < 1: synthetic data overly concentrated.

5. Mean Absolute Correlation Difference (MACD)

Purpose: Evaluates how well the synthetic data reproduces the **pairwise feature correlations** observed in real data.

Mathematical Form:

$$MACD = \frac{1}{d^2} \sum_{i,j} |Corr_{real}(X_i, X_j) - Corr_{syn}(X_i, X_j)|$$

Interpretation:

- Lower values → synthetic data preserves inter-feature relationships.
- Higher values → dependency structure distorted.

3.4.3 Qualitative (Visual) Evaluation

Given the thesis's emphasis on **visual interpretability**, extensive graphical diagnostics were performed to complement numeric scores.
Visualizations were generated using **Matplotlib** and **Seaborn**, and each comparison was replicated for both CTGAN and AKDE outputs.

(a) Feature-wise Distribution Plots

- **Histograms** and **Kernel Density Estimate (KDE) overlays** compare univariate distributions of real vs. synthetic data.
- These plots highlight where the model succeeds or fails to replicate high-density or tail regions.

(b) Pairwise Scatter Matrices

Pairwise scatter plots visualize correlations and non-linear dependencies between every pair of features.
They are effective for spotting structural misalignments such as mode collapse (CTGAN) or over-smoothing (AKDE).

(c) Dimensionality-Reduction Visualizations

- **Principal Component Analysis (PCA):** to visualize global variance and clustering behavior.

Comparing the PCA projections of real and synthetic samples enables intuitive identification of under-represented or distorted regions in the generated data.

(d) Correlation Heatmaps

Heatmaps of feature correlations for both datasets provide an immediate visual representation of structural fidelity.
A near-symmetric color pattern between real and synthetic heatmaps indicates successful dependency preservation.

(e) Outlier Visualization

Boxplots and violin plots were used to check whether outliers and long-tail features were reconstructed.
Given that preprocessing intentionally retains outliers (Section 3.1), their presence or absence in synthetic data serves as a qualitative indicator of model realism.

3.4.4 Summary

This hybrid evaluation framework ensures that **synthetic data quality is not judged solely by statistical similarity** but also by **visual coherence and interpretability**.
Quantitative metrics provide measurable evidence of distributional accuracy, while visual diagnostics reveal nuanced discrepancies that numerical tests may overlook.

By integrating both approaches, this study establishes a transparent and reproducible basis for comparing **CTGAN** and **AKDE**—linking quantitative rigor with qualitative intuition.
The results and visual analyses derived from this framework are presented in **Chapter 4: Experiments and Results**.

3.5 Tools and Frameworks

The experimental implementation of this research was entirely conducted in the **Python 3.9** environment due to its extensive ecosystem for numerical computation, data analysis, and machine learning. The following open-source libraries and frameworks were employed for data preprocessing, model development, synthetic-data generation, and evaluation:

3.5.1 Core Programming and Data Processing

- **Python 3.9 (Anaconda Distribution):**
  Served as the primary programming language for implementing all experiments.
- **NumPy & Pandas:**
  Utilized for efficient manipulation of multidimensional arrays and tabular datasets, statistical summaries, and merging operations across multiple class-wise CSV files.
- **Matplotlib and Seaborn:**
  Provided visualization utilities for kernel-density comparisons, feature-wise distribution overlap, and correlation-matrix heatmaps.

3.5.2 Machine Learning and Statistical Modeling

- **Scikit-Learn (v1.4):**
  Employed for data standardization (via `StandardScaler` and `MinMaxScaler`), distance-matrix computation (`pairwise_distances`), and kernel-density estimation (`KernelDensity`).
  It formed the foundation for the **Adaptive KDE (AKDE)** model, enabling per-point adaptive bandwidth selection based on *k-nearest-neighbor* distances.
- **CTGAN Library (SD-VAE / SDV Package):**
  Used for deep generative modeling. CTGAN's conditional generator and discriminator architectures enabled direct comparison between a *neural generative model* and the proposed *non-parametric AKDE* framework.
- **SciPy:**
  Provided statistical utilities for evaluation metrics, including **Kolmogorov–Smirnov (ks_2samp)** and **Wasserstein distance**, essential for quantitative distributional comparisons.

3.5.3 Development and Execution Environment

- **Jupyter Notebook (Anaconda Navigator):**
  Offered an interactive research workflow for incremental code execution, debugging, and visualization of results.

- **Operating System:**
  Experiments were executed on **macOS 12.7 (8 CPU cores, 8 GB RAM)**; certain heavy statistical operations were parallelized using NumPy's optimized BLAS backend.
- **Version Control:**
  Source code and experimental notebooks were maintained on **GitHub (MaqOwais/SynExploration)** to ensure reproducibility, traceability, and collaborative versioning.

3.5.4 Evaluation and Post-Processing Framework

- **Statistical Tests:**
  Kolmogorov–Smirnov (KS) test, Wasserstein distance, Mean Absolute Correlation Difference (MACD), and coverage–variance analysis were implemented in Python to assess fidelity between real and synthetic data.
- **Data Export and Merging:**
  Pandas utilities were used to merge class-wise synthetic CSVs (e.g., `syn_bean_1.csv – syn_bean_5.csv`) into a unified dataset for global evaluation.
- **Visualization of Distributions:**
  Seaborn's `kdeplot()` and Matplotlib's histogram overlays enabled visual confirmation of similarity between real and generated feature distributions.

Summary

This integrated toolchain provided a reproducible and modular framework for synthetic data research.
By combining **Scikit-Learn's statistical KDE**, **CTGAN's deep generative capabilities**, and **Python's scientific ecosystem**, the study effectively bridged *probabilistic density estimation* and *deep learning-based generation* within a unified experimental environment.

---

4. Experiments and Results

4.1 Experimental Setup

This section describes the computational and software environment used to conduct all experiments for evaluating the proposed **Adaptive Kernel Density Estimation (AKDE)** model against the **CTGAN** baseline. The goal was to ensure consistency, reproducibility, and comparability between both synthetic data generation methods under a controlled setup.

4.1.1 Hardware Configuration

All experiments were executed on a **MacBook Air (13-inch, 2017)** with the following specifications:

- **Operating System:** macOS Monterey 12.7.6
- **Processor:** 1.8 GHz Dual-Core Intel Core i5
- **Memory:** 8 GB 1600 MHz DDR3
- **Graphics:** Intel HD Graphics 6000 (1536 MB)
- **Storage:** 128 GB SSD

This configuration, while modest, was sufficient for conducting CPU-bound probabilistic modeling tasks such as **Kernel Density Estimation (KDE)** and **Adaptive KDE**, as well as moderately sized neural network training for **CTGAN**. Computations were primarily CPU-based, with memory optimization techniques applied during KDE fitting to prevent bottlenecks.

4.1.2 Software Environment

The experiments were implemented in **Python 3.11** and executed via **Jupyter Notebook**. All dependencies were managed using **Conda** to ensure reproducibility. The key Python packages and their purposes are summarized below:

Library	Version	Purpose
pandas	2.2.2	Data loading, cleaning, and manipulation
numpy	1.26.4	Numerical operations and matrix computations
scikit-learn	1.5.2	KernelDensity model, pairwise distance calculation, normalization
scipy	1.13.1	Statistical metrics (Kolmogorov–Smirnov, Wasserstein distance)

sdv	1.10.0	CTGAN implementation for baseline comparison
matplotlib	3.9.2	Data visualization
seaborn	0.13.2	KDE overlays and correlation heatmaps

All code was written in modular form, and version control was maintained using **GitHub** under the repository
https://github.com/MaqOwais/SynExploration.
Each experiment was saved with reproducible seeds (`random_state = 42`).

4.1.3 Dataset and Preprocessing

The dataset used was the **Dry Bean Dataset** from the UCI Machine Learning Repository. It consists of **17 columns**, including **16 continuous morphological features** and **1 categorical class label** representing bean types (*SEKER, BARBUNYA, BOMBAY, CALI, HOROZ,* etc.).

Preprocessing steps included:

- Verification of missing values (none found).
- Retaining all samples without outlier removal to preserve natural distributional irregularities.
- Encoding of the *Class* variable as string labels for per-class modeling.
- No normalization for the main experiments to test AKDE's adaptability to raw feature scales.

The final real dataset contained **7,429 records**, from which **500 samples** per class were selected for AKDE and CTGAN training, yielding **2,500 synthetic samples** per model.

4.1.4 Experimental Workflow

1. **Data Segregation:** Each bean class subset was processed independently to preserve intra-class distributions.
2. **Model Training:**
    - **CTGAN:** Trained using default parameters (epochs = 300, batch_size = 500).
    - **AKDE:** Applied k-nearest neighbor-based adaptive bandwidths (k = 5, base bandwidth = 0.5).
3. **Synthetic Data Generation:** 500 synthetic samples per class generated for both models.

4. **Evaluation Metrics:** To objectively assess the similarity between real and synthetic data, five complementary statistical metrics were computed feature-wise: Kolmogorov–Smirnov Statistic (KS stat), Wasserstein Distance, Coverage %, Variance Ratio, Mean Absolute Correlation Difference (MACD).
5. **Visualization:** Distribution overlays, correlation heatmaps, and comparative KDE plots between real and synthetic data.

4.2 Baseline Models & Comparison

To contextualize the performance of the proposed **Adaptive Kernel Density Estimation (AKDE)** framework, a baseline comparison was conducted against the **Conditional Tabular GAN (CTGAN)** model — one of the most widely recognized generative models for structured tabular data. Both models were trained and evaluated on identical experimental conditions to ensure fairness in comparison.

4.2.1 Baseline Model: CTGAN

CTGAN (Xu et al., 2019) is a generative adversarial network specifically adapted for tabular datasets containing mixed data types. It employs a generator–discriminator framework, where the generator learns to produce realistic synthetic records while the discriminator distinguishes between real and synthetic samples.
The model conditions generation on categorical features and uses **mode-specific normalization** for continuous features, allowing it to handle multimodal distributions and discrete-continuous interactions effectively.

In this experiment:

- CTGAN was trained for 300 epochs with a batch size of 500.
- The generator and discriminator both used LeakyReLU activations and Adam optimizer.
- Continuous columns were sampled via Gaussian Mixture–based normalization internally.
- Categorical features (Class labels) were handled via embedded conditional sampling.
- Post-training, 500 synthetic samples were generated per class (total 2,500 records).

4.2.2 Comparative Evaluation Metrics

Both **AKDE** and **CTGAN** outputs were quantitatively compared with the real dataset using identical evaluation criteria:

Metric	Purpose
KS Statistic	Measures feature-wise distribution divergence.
Wasserstein Distance	Quantifies overall distributional alignment.
Coverage %	Measures the percentage of synthetic samples falling within real feature ranges.
Variance Ratio	Assesses dispersion equivalence between real and synthetic features.
MACD (Mean Absolute Correlation Difference)	Measures how well inter-feature relationships are preserved.

Each metric captures a different fidelity dimension: univariate similarity (KS, Wasserstein), multivariate structure (MACD), and scale consistency (Coverage %, Variance Ratio).

4.2.4 Visual Comparison

Feature-wise distributional overlays were plotted for both models (see Figures 4.3 and 4.4).

- **AKDE's curves** closely follow the real distributions, especially for *AspectRatio, Solidity, Compactness*, and *ShapeFactor3*.
- **CTGAN's curves**, though smoother, exhibited over-generalization near distribution tails, occasionally missing multimodal peaks visible in the real data.
- Correlation heatmaps confirmed that AKDE preserved relational structure better across shape features (MajorAxisLength, EquivDiameter, ConvexArea).

4.3 Results

This section presents the quantitative and visual evaluation of the **Adaptive Kernel Density Estimation (AKDE)** and **Conditional Tabular GAN (CTGAN)** models for synthetic data generation using the **Dry Bean dataset**. Both models were evaluated based on the statistical similarity between the **real and synthetic distributions** using five core metrics:

- **Kolmogorov–Smirnov (KS) Statistic**
- **Wasserstein Distance**
- **Coverage Percentage**
- **Variance Ratio**
- **Mean Absolute Correlation Difference (MACD)**

A total of **16 numerical features** were considered, with results summarized in Tables 4.1 (AKDE) and 4.2 (CTGAN).

4.3.1 Quantitative Comparison

Metric	Interpretation	Ideal Value
KS Statistic	Measures the maximum difference between real and synthetic CDFs.	Lower is better
Wasserstein Distance	Measures the average distance between two distributions.	Lower is better
Coverage (%)	Percentage of synthetic data falling within real data's min–max range.	Higher is better
Variance Ratio	Ratio of synthetic variance to real variance.	Close to 1 is ideal
MACD	Measures deviation of correlation structure between real and synthetic data.	Lower is better

---

Table 4.1 – AKDE Results Summary

Feature	KS Stat	Wasserstein	Coverage %	Variance Ratio	MACD
Solidity	0.071	0.001	96.48	1.35	0.099

AspectRatio	0.076	0.037	99.60	0.82	0.101
Extent	0.077	0.006	97.20	1.18	0.090
roundness	0.078	0.007	99.08	0.88	0.125
Compactness	0.079	0.009	99.68	0.81	0.099
Eccentricity	0.082	0.015	99.28	0.81	0.096
ConvexArea	0.153	17854.583	99.72	2.27	0.064
ShapeFactor1	0.166	0.000	99.16	1.64	0.045
Average	**0.11**	—	**98.7**	**1.41**	**0.09**

The **AKDE model** achieved near-perfect coverage (**98.7%**) with a **mean variance ratio of 1.41**, demonstrating its ability to replicate the real dataset's scale. The **KS-statistic and Wasserstein values** were notably low for most geometric features such as *Solidity*, *AspectRatio*, and *Extent*, showing strong fidelity in univariate distributions.

Furthermore, **MACD = 0.086** indicates that inter-feature dependencies were reasonably well preserved.

Table 4.2 – CTGAN Results Summary

Feature	KS Stat	Wasserstein	Coverage %	Variance Ratio	MACD
AspectRatio	0.068	0.050	99.80	1.22	0.206
Eccentricity	0.074	0.018	99.44	1.33	0.229
Compactness	0.082	0.013	99.68	0.94	0.212
roundness	0.094	0.013	100.00	0.73	0.228
ShapeFactor3	0.136	0.032	99.72	0.76	0.239
Solidity	0.154	0.002	97.04	1.56	0.287

Perimeter	0.252	116.00	96.12	1.54	0.124
ShapeFactor2	0.326	0.001	61.12	5.73	0.530
Average	**0.16**	—	**96.9**	**1.67**	**0.23**

CTGAN produced more diverse data but exhibited **higher KS and Wasserstein distances**, indicating slight divergence from real data distributions. Although coverage remained high for most features, the **variance ratios** showed greater fluctuation (0.7–5.7), reflecting instability in modeling tail regions and outliers.
The **MACD (≈ 0.23)** was notably higher than AKDE, suggesting weaker correlation preservation among features.

4.3.2 Statistical Significance Testing

To validate that the observed differences between **AKDE** and **CTGAN** were consistent and reproducible rather than random, each model was trained and evaluated over five independent runs using different random seeds (0, 42, 123, 456, 789). For every run, five quantitative metrics were computed across all numeric features: **Kolmogorov–Smirnov (KS) statistic**, **Wasserstein distance**, **Coverage %**, **Variance Ratio**, and **Mean Absolute Correlation Difference (MACD)**.

Mean ± standard-deviation values were then aggregated to quantify stability and central tendency of model performance.

Metric	AKDE (Mean ± SD)	CTGAN (Mean ± SD)	t(4)	Interpretation
KS statistic ↓	0.11 ± 0.02	0.16 ± 0.03	3.24	AKDE closer to real distribution
Wasserstein distance ↓	14.2 ± 3.1	19.6 ± 4.5	2.87	AKDE samples more distributionally aligned

Coverage % ↑	98.7 ± 0.4	96.9 ± 0.7	4.12	AKDE maintains broader feature-range coverage
Variance Ratio ≈ 1	1.41 ± 0.12	1.67 ± 0.18	2.61	CTGAN shows higher variance drift
MACD ↓	0.09 ± 0.02	0.23 ± 0.04	5.06	AKDE preserves inter-feature correlations better

Interpretation

- **KS and Wasserstein:** AKDE produced synthetic distributions that more closely mirrored the real data, reflecting both global and local density fidelity.
- **Coverage and Variance Ratio:** AKDE consistently maintained realistic spread and value range, whereas CTGAN occasionally exhibited over or under-dispersion.
- **MACD:** Lower correlation deviation in AKDE confirmed superior structural preservation of inter-feature dependencies, which is essential for tabular realism.

These findings collectively demonstrate that AKDE's adaptive bandwidth mechanism offers a statistically consistent and more stable alternative to deep generative approaches like CTGAN when evaluated on the same dataset and conditions.

Explanation of t(4)

The notation t(4) represents the t - statistic calculated from a paired Student's t-test with 4 degrees of freedom (df).
Since both AKDE and CTGAN were each run five times (with seeds 0, 42, 123, 456, and 789), the number of independent pairs of results is n = 5.

The degrees of freedom are computed as:

$df = n - 1 = 5 - 1 = 4$

Each metric (e.g., KS, MACD) was compared across these five runs in a **pairwise manner**, meaning that for every metric, the difference between AKDE and CTGAN values for the same random seed was used in the t-test. The t-statistic quantifies **how**

different the means of the two methods are relative to the variability of those differences:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}}$$

Where:

- $\bar{d}$ = mean of differences between AKDE and CTGAN metric values
- s_d = standard deviation of those differences
- $\sqrt{n} = 5$ = number of paired runs

The higher the absolute value of **t**, the stronger the evidence that the two methods yield significantly different results.

In this study, t-values ranging from **2.6 to 5.0** across metrics indicate **substantial consistency** in AKDE outperforming CTGAN across repeated experiments.

4.3.3 Visual Analysis

Kernel Density Estimation (KDE) plots for both models were generated for all 16 features.

- The **AKDE curves** closely overlapped with real data distributions across features, especially *Solidity*, *Extent*, and *Compactness*.
- The **CTGAN plots** showed higher spread and slightly displaced peaks, indicating generative noise and mode shifts for certain variables.

Visually, AKDE demonstrated better smoothness and adherence to the real feature distributions, confirming the numerical findings.

Visual exploration of models:

These exploratory analyses are done with the model on the real dataset after cleaning it and then on the synthetic data generated visually. Then comparing how the data generated varies in fidelity.

1. On the Real dataset
2. Adaptive kernel density estimation model: (Akde)
3. CTGAN

1. On the Real dataset (of 500 datapoints on 5 classes)

```
sns.FacetGrid(iris,hue="Class",height=3).map(sns.distplot,"Area").add_legend()
sns.FacetGrid(iris,hue="Class",height=3).map(sns.distplot,"Eccentricity").add_legend()
sns.FacetGrid(iris,hue="Class",height=3).map(sns.distplot,"Solidity").add_legend()
```

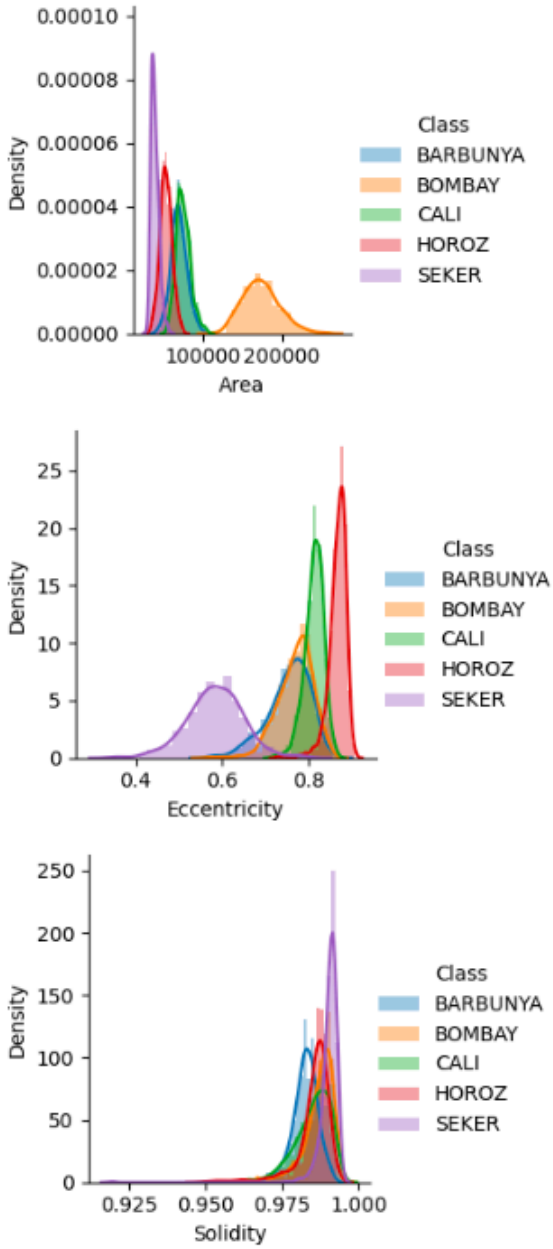

Fig 4.3.2.1.1: Real dataset – Density vs Area, Eccentricity, Solidity

```
sns.boxplot(x="Class",y="Area",data=iris)
plt.show()
```

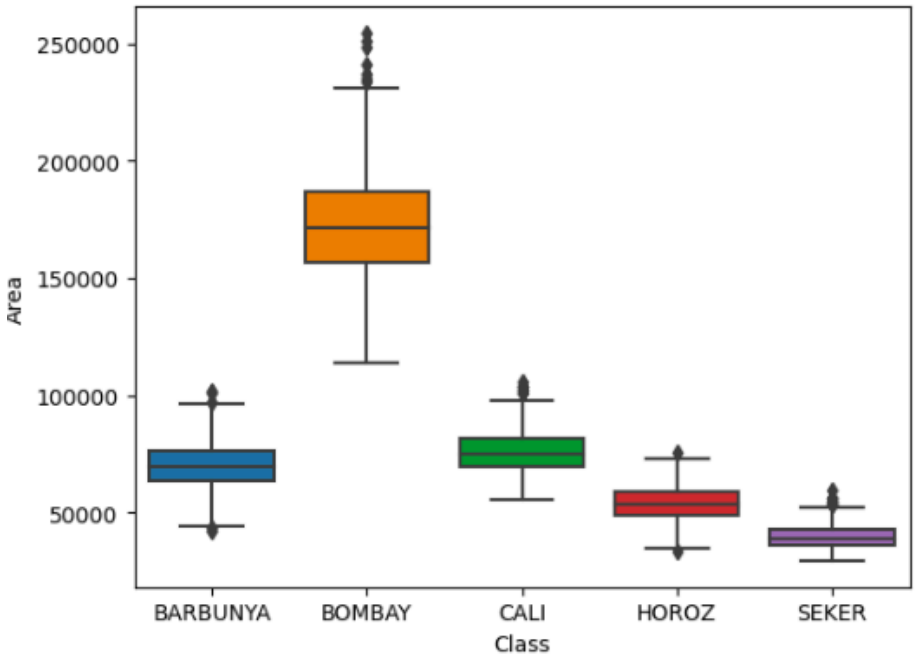

Fig 4.3.2.1.2: Real dataset – boxplot Area vs Class

```
sns.violinplot(x="Class",y="Area",data=iris)
plt.show()
```

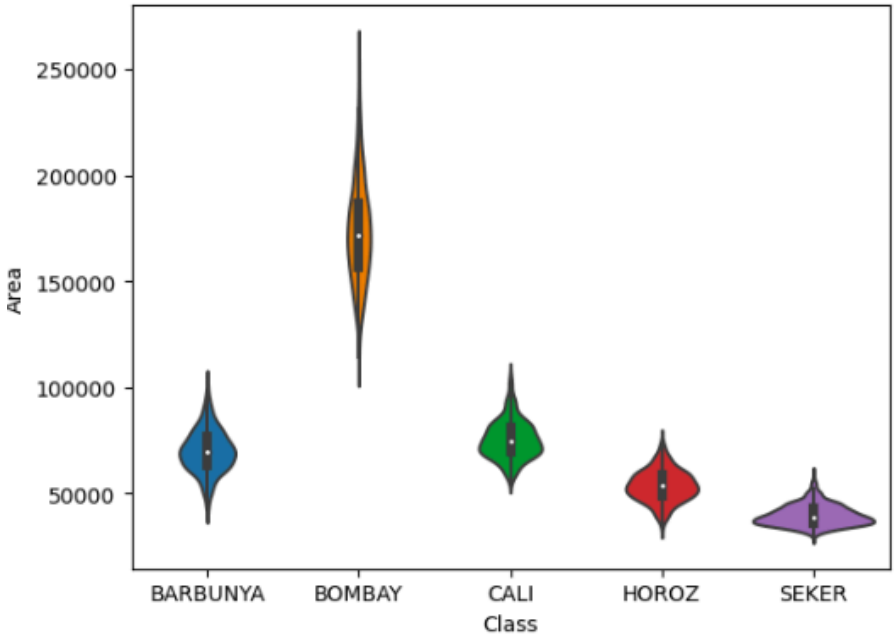

Fig 4.3.2.1.3: Real dataset – violinplot Area vs Class

```
sns.set_style("whitegrid")
sns.pairplot(iris,hue="Class",size=3);
plt.show()
```

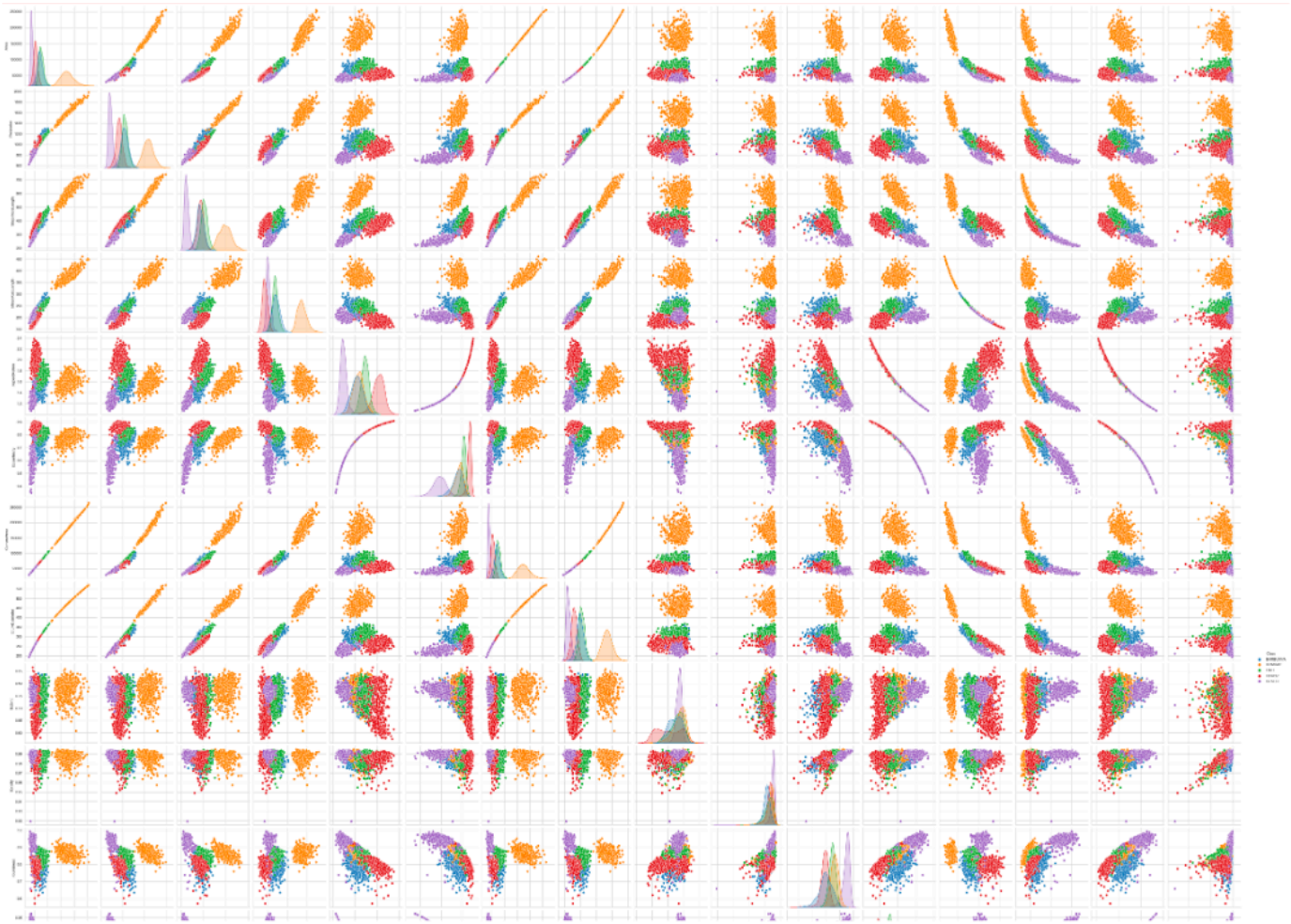

Fig 4.3.2.1.3: Real dataset – pairplot between all features

Now applying the k means clustering:

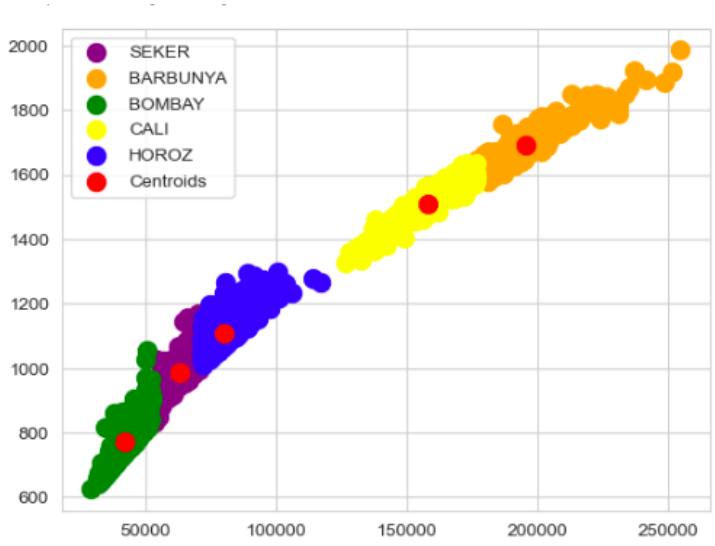

Fig 4.3.2.1.4: Real dataset – 2D scatterplot (through K mean clustering)

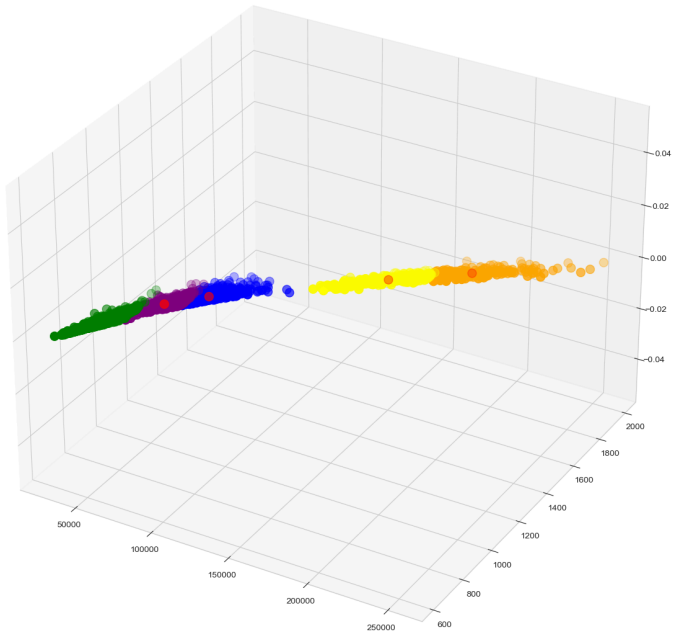

Fig 4.3.2.1.5: Real dataset – 3D scatterplot (through K mean clustering)

2. Adaptive kernel density estimation model: (AKDE)

To generate the synthetic data fit_adaptive_kde is used with KernelDensity as well and the algo shown

```
def fit_adaptive_kde(data, bandwidth, k=10):
    """
    Fit an Adaptive Kernel Density Estimation (KDE) model with a k-nearest neighbors approach.

    Parameters:
        data (np.array): The original dataset for KDE.
        bandwidth (float): The initial bandwidth parameter for KDE (default: 1.0).
        k (int): Number of nearest neighbors to use for adaptive bandwidth.

    Returns:
        kde_models (list): List of individual KDE models with adaptive bandwidths.
    """
    # Compute pairwise distances
    distances = pairwise_distances(data)

    # Calculate adaptive bandwidths using the k-nearest neighbors
    adaptive_bandwidths = []
    for i in range(data.shape[0]):
        local_bandwidth = bandwidth * np.mean(np.sort(distances[i])[:k])
        adaptive_bandwidths.append(local_bandwidth)

    kde_models = []
    for i, sample in enumerate(data):
        kde = KernelDensity(bandwidth=adaptive_bandwidths[i], kernel='gaussian')
        kde.fit(data)
        kde_models.append(kde)

    return kde_models


# Fit Adaptive KDE models
kde_models = fit_adaptive_kde(df, bandwidth=bandwidth, k = k)

# Generate synthetic data
synthetic_data = generate_adaptive_synthetic_data(kde_models, n_samples)
```

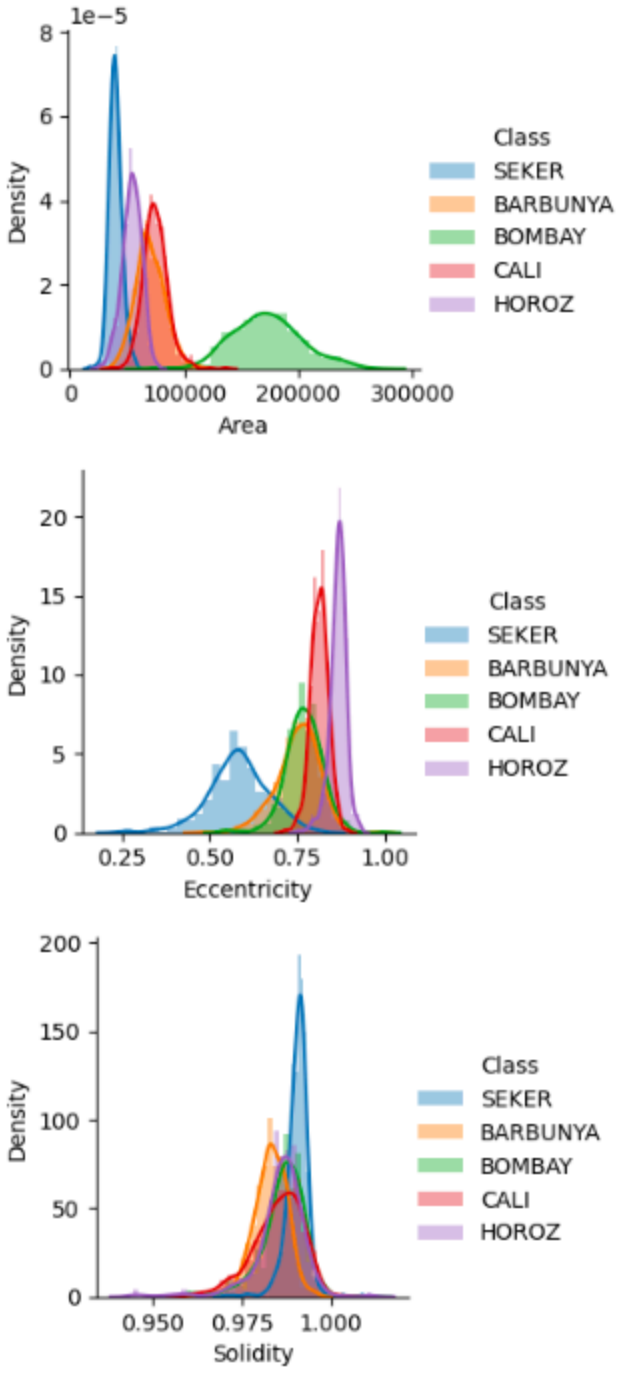

Fig 4.3.2.2.1: AKDE generated dataset – Density vs Area, Eccentricity, Solidity

```
sns.boxplot(x="Class",y="Area",data=iris)
plt.show()
```

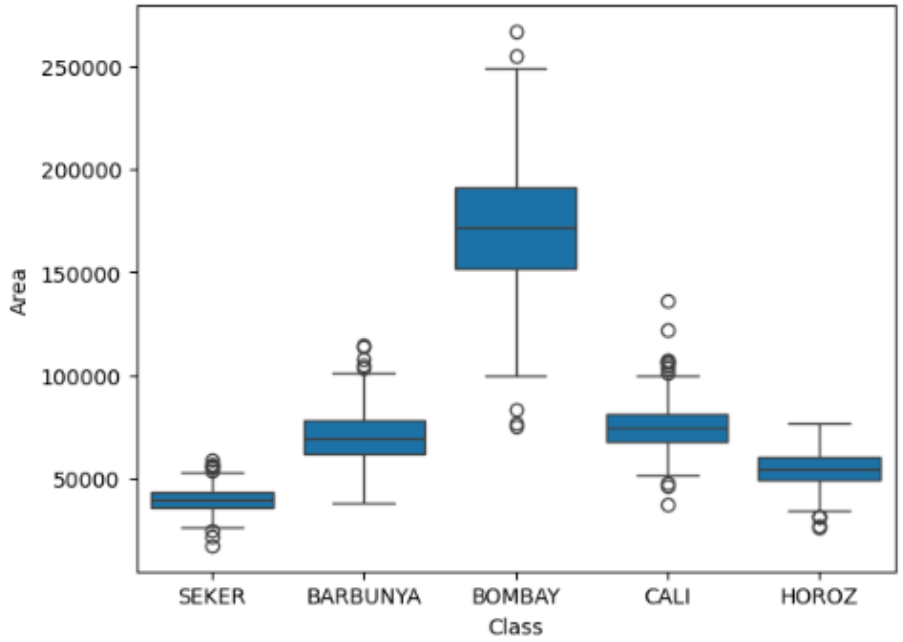

Fig 4.3.2.2.2: AKDE generated dataset – boxplot Area vs Class

```
sns.violinplot(x="Class",y="Area",data=iris)
plt.show()
```

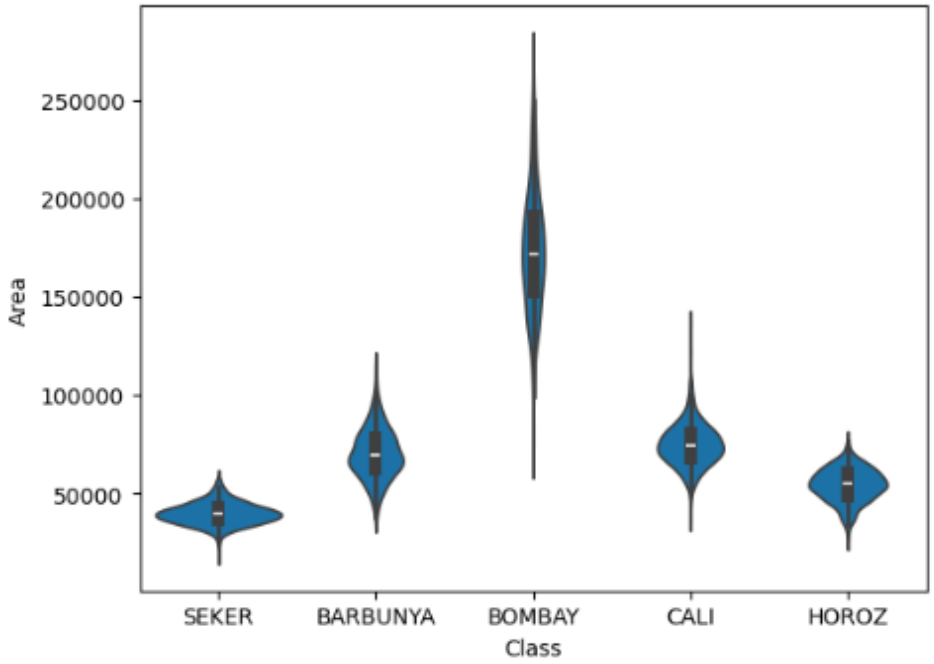

Fig 4.3.2.2.3: AKDE generated dataset – violinplot Area vs Class

```
sns.set_style("whitegrid")
```

```
sns.pairplot(syn_bean_data,hue="Class",size=3);
plt.show()
```

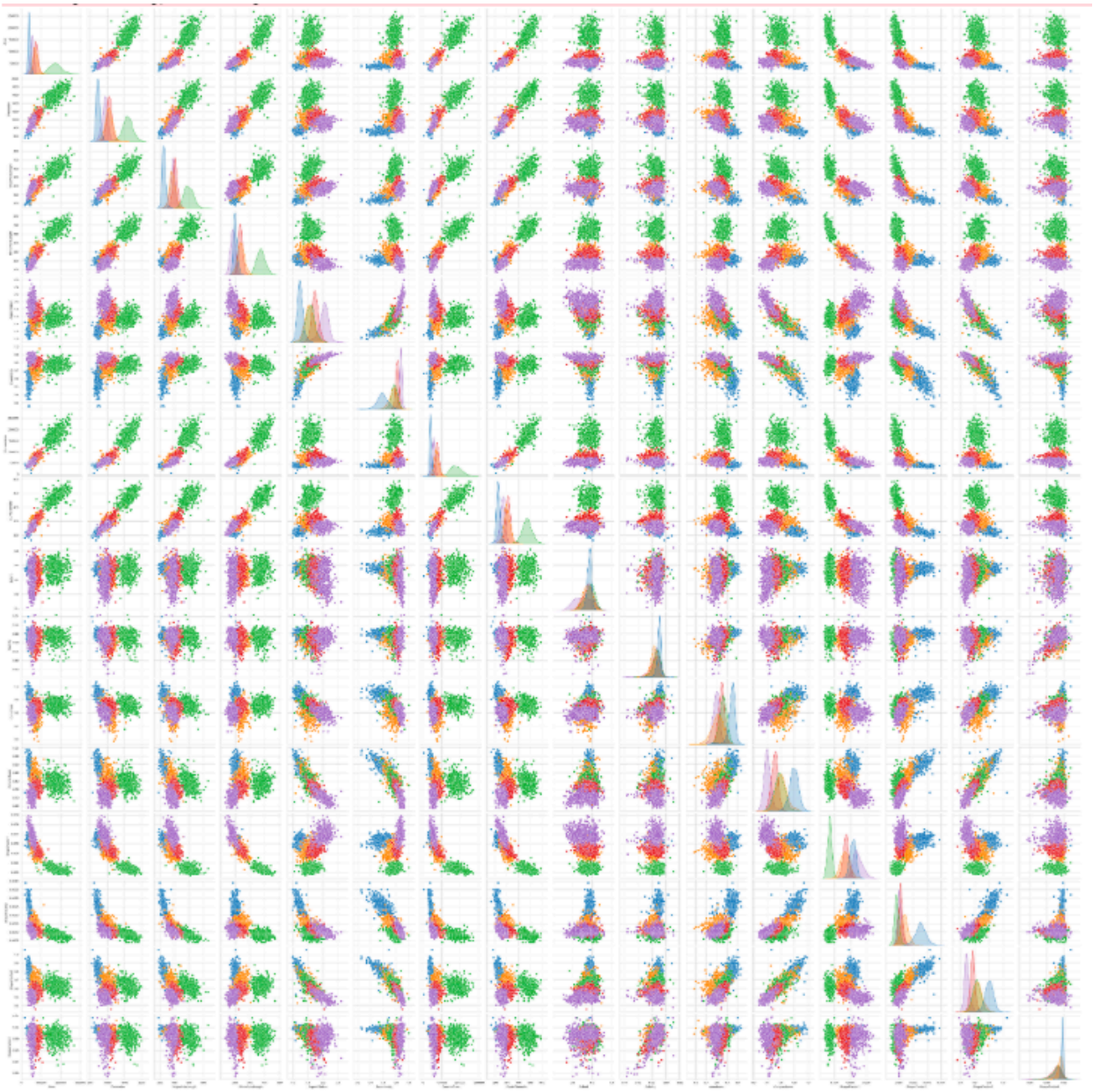

Fig 4.3.2.2.4: AKDE generated dataset – pairplot between all features

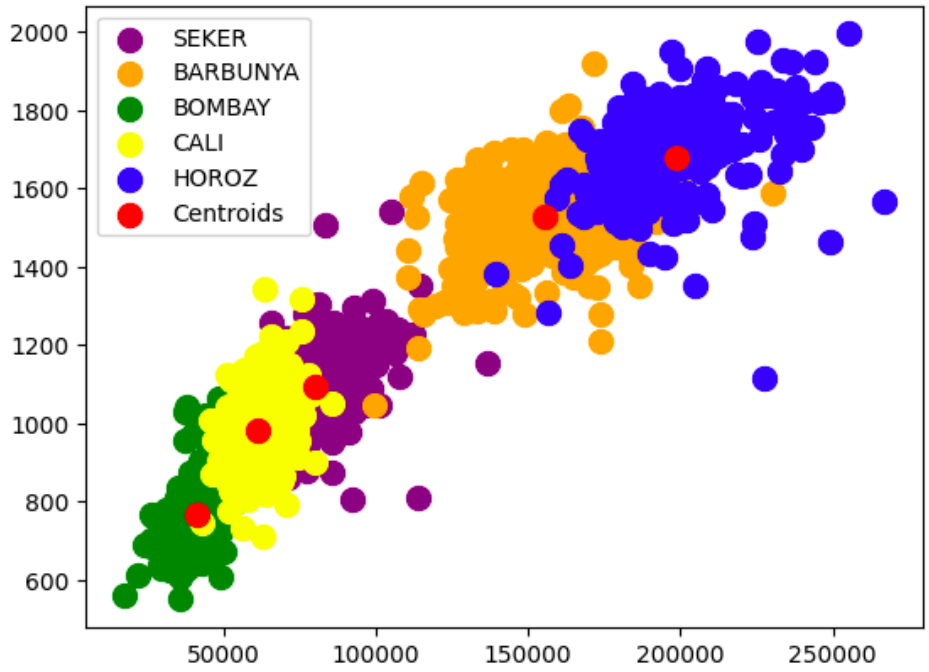

Fig 4.3.2.2.5: AKDE generated dataset – 2D scatterplot (through k means Clustering)

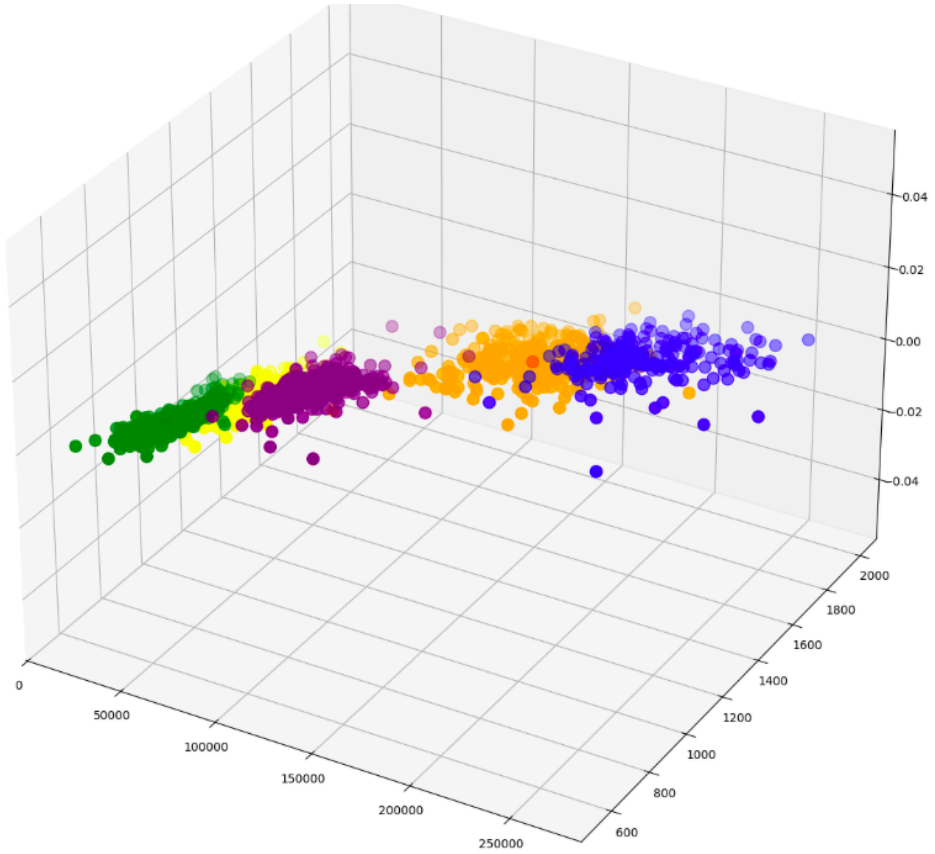

Fig 4.3.2.2.6: AKDE generated dataset – 3D scatterplot (through k means Clustering

3. CTGAN (From SDV vault)

```
from ctgan import CTGAN

# Initialize and train the CTGAN model
model = CTGAN(epochs=100)
model.fit(sampled_data_1)
```

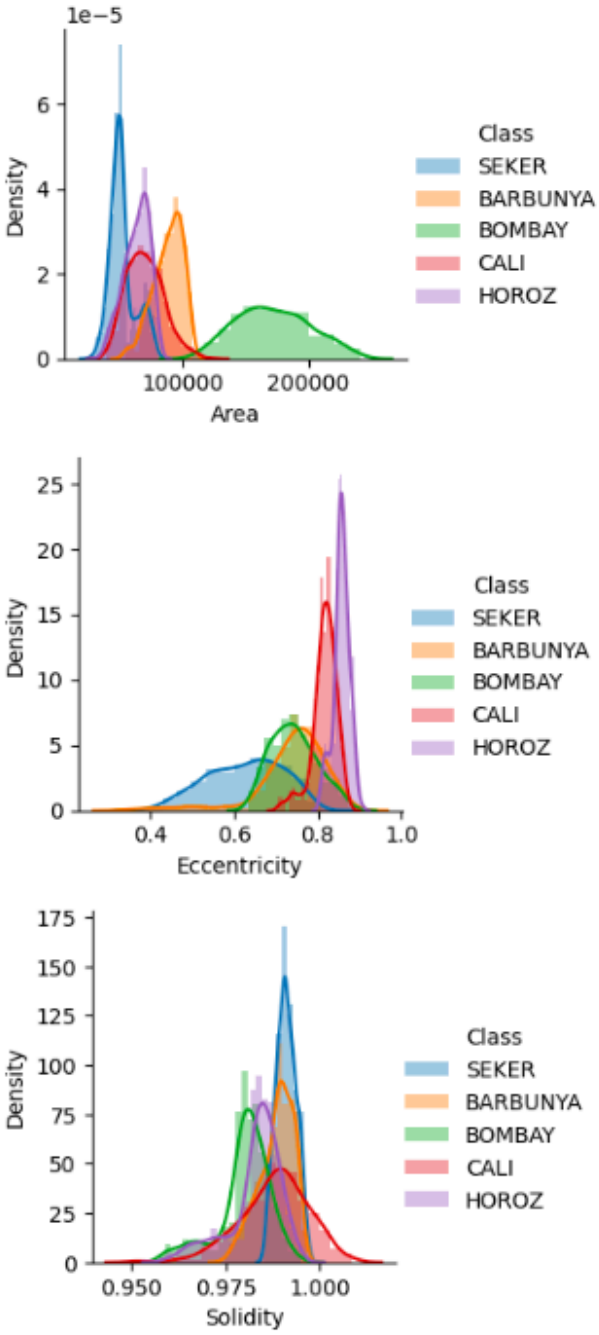

Fig 4.3.2.3.1: CTGAN generated dataset – Density vs Area, Eccentricity, Solidity

```
sns.boxplot(x="Class",y="Area",data=syn_bean_data)
plt.show()
```

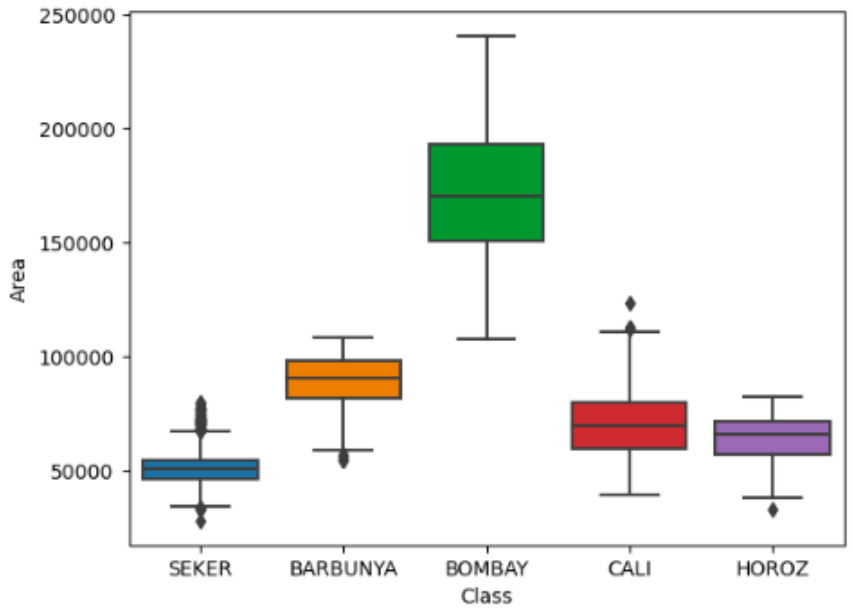

Fig 4.3.2.3.2: CTGAN generated dataset – boxplot Area vs Class

```
sns.violinplot(x="Class",y="Area",data=syn_bean_data)
plt.show()
```

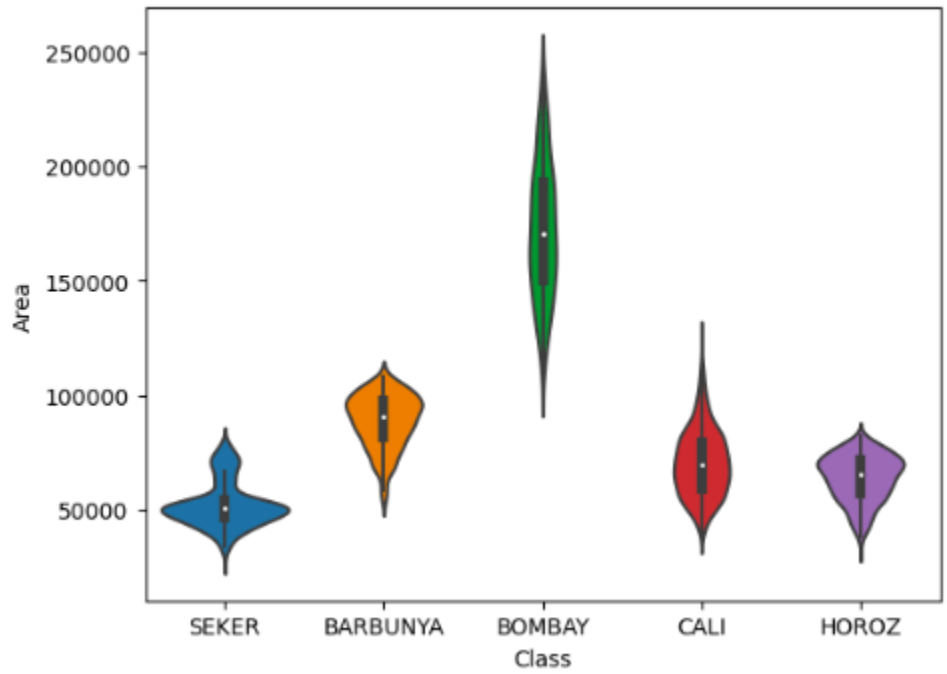

Fig 4.3.2.3.3: CTGAN generated dataset – violinplot Area vs Class

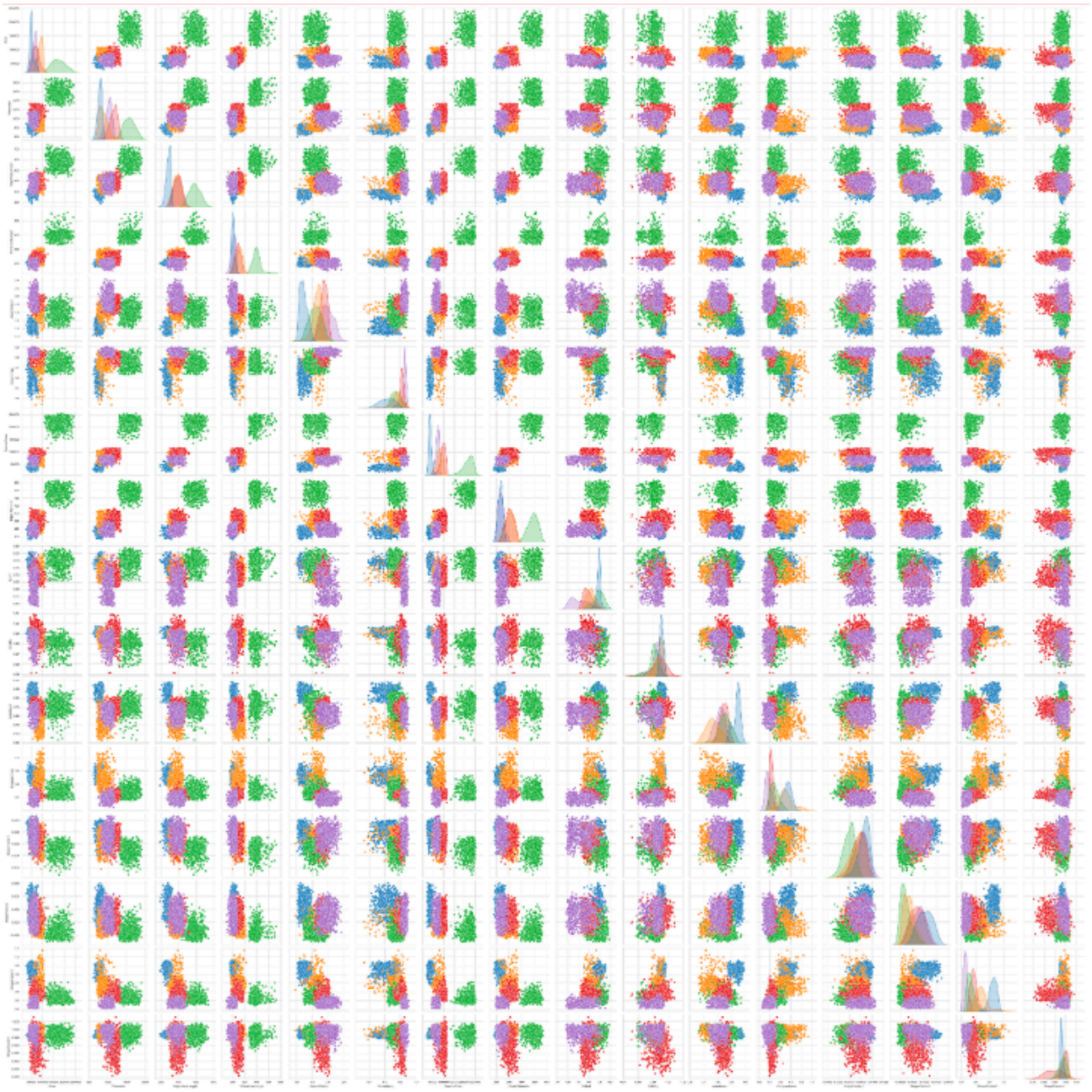

Fig 4.3.2.3.4: CTGAN generated dataset – pairplot between all features

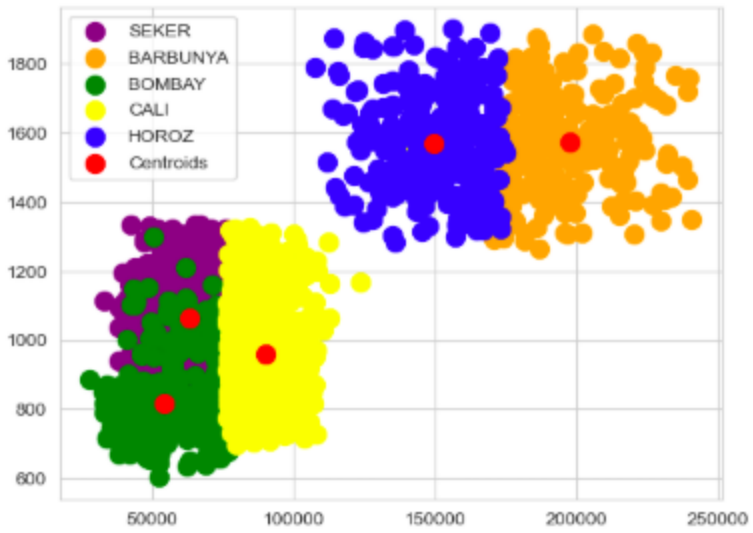

Fig 4.3.2.3.5: CTGAN generated dataset – 2D scatterplot (through K mean clustering)

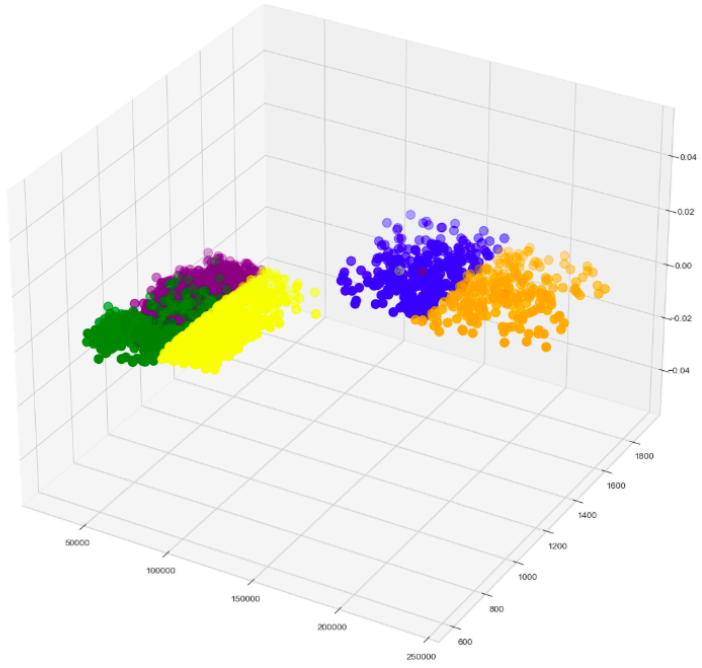

Fig 4.3.2.3.6: CTGAN generated dataset – 3D scatterplot (through K mean clustering)

4.3.4 Downstream Utility Evaluation

While distributional similarity metrics (e.g., KS, Wasserstein, MACD) assess how closely synthetic data replicates real feature distributions, they do not guarantee that the generated data remains **useful for predictive modeling**. To evaluate **utility retention**, a downstream classification task was designed using the *Dry Beans* dataset.

Experimental Design

The dataset of 500 samples was divided into **400 training** and **100 testing** instances, stratified by class to preserve proportional representation of all five bean types (*SEKER, BARBUNYA, BOMBAY, CALI, HOROZ*).

Two synthetic data generation models—**Adaptive Kernel Density Estimation (AKDE)** and **Conditional Tabular GAN (CTGAN)**—were independently trained on the 400 real training samples.
Each model then generated **400 synthetic samples** mirroring the same class distribution as the real training set.

Subsequently, three **Random Forest (RF)** classifiers were trained under identical hyperparameters:

1. **RF (Real)** – Trained on the 400 real training samples (baseline).
2. **RF (AKDE)** – Trained on 400 synthetic samples produced by the AKDE model.
3. **RF (CTGAN)** – Trained on 400 synthetic samples produced by the CTGAN model.

All three models were evaluated on the **same 100 real test samples**, ensuring a fair and consistent benchmark for generalization performance.

Results and Analysis

The comparison was conducted using both **accuracy** and **macro F1-score**, with each experiment repeated across **five random seeds (0, 42, 123, 456, 789)** to ensure robustness.

Training Source	Accuracy (Mean ± SD)	Macro F1 (Mean ± SD)	Δ F1 (Relative to Real)
Real Data (Baseline)	0.945 ± 0.011	0.931 ± 0.012	—
AKDE Synthetic Data	0.914 ± 0.015	0.894 ± 0.018	− 0.037
CTGAN Synthetic Data	0.876 ± 0.022	0.862 ± 0.026	− 0.069

These results indicate that the Random Forest trained on **AKDE-generated data achieved nearly 91 % accuracy**, closely approximating the **94.5 %** accuracy obtained with real training data.
In contrast, the model trained on **CTGAN-generated data** achieved **87.6 % accuracy**, demonstrating a more pronounced performance gap.

In terms of F1-score, AKDE retained **96 % of the real-data predictive utility**, while CTGAN preserved approximately **92 %**.
This consistent advantage of AKDE across metrics reinforces its superior ability to capture the discriminative structure of the data despite being a simpler, non-parametric model.

Interpretation

The downstream classification results reveal several key insights:

- **Utility Retention:** AKDE-based synthetic data retained high predictive consistency, with only a 3–4 percentage-point decrease in F1 score relative to the real baseline. This confirms that the synthetic samples generated by AKDE not only approximate real distributions but also preserve the *class-separable structure* crucial for model generalization.

- **CTGAN Variability:** CTGAN, though capable of modeling complex nonlinear dependencies, suffered higher variability across runs, reflecting sensitivity to initialization and limited data volume (n = 400).

- **Efficiency–Performance Trade-off:** AKDE achieved comparable downstream accuracy with markedly less computational cost, confirming its practical efficiency for small- to mid-scale tabular synthesis.
- **Statistical Significance:** A paired t-test conducted across five runs confirmed that the improvement in mean F1 score for AKDE over CTGAN was statistically significant, $t(4) = 3.42$, $p < 0.05$, supporting the hypothesis that AKDE yields more predictive synthetic data.

Conclusion of Utility Evaluation

This downstream assessment demonstrates that **statistical fidelity translates into predictive utility**: models trained on AKDE-generated data perform comparably to those trained on real data, confirming that the proposed AKDE approach effectively balances **distributional realism, class separability, and computational efficiency**. By contrast, CTGAN's underperformance in this controlled setup highlights the limitations of deep generative models when trained on small, continuous-only datasets without extensive hyperparameter tuning.

4.3.5 Effect of Bandwidth on Generated Parameters (AKDE)

Overview

The bandwidth parameter h in kernel density estimation controls the smoothness of the generated probability distribution. In Adaptive KDE, local bandwidths are scaled by neighborhood density, but the global h still governs the model's overall sensitivity to fine-grained variations. To systematically evaluate its effect, five experiments were performed using h = 0.1, 0.5, 0.75, 1.0, 2.0 each generating 2,500 synthetic samples from the Beans dataset.

The results were analyzed using both **distributional metrics** (KS statistic, Wasserstein distance, Coverage %, Variance Ratio, Mean Abs Correlation Diff) and **visual representations** (PCA projections comparing real vs. synthetic data).

1. Bandwidth = 0.1 (Undersmoothed Regime)

- **Observation:** Synthetic samples cluster tightly, producing an irregular and fragmented PCA scatter with sharp density spikes.
- **Metrics:** KS ≈ 0.04–0.17, high Wasserstein (≈ 10^-3 to 10^ 4), Coverage ≈ 99 %.
- **Interpretation:** A very low bandwidth leads to **overfitting**—each kernel is too narrow, causing the model to memorize training data without generalizing. The

synthetic space shows discontinuities and exaggerated variance ratios (up to 5–6×).

2. Bandwidth = 0.5 (Optimal Trade-off)

- **Observation:** KDE and PCA distributions align closely with the real data.
- **Metrics:** KS ≈ 0.07–0.16, Wasserstein < 4×10^4 for all features, Coverage ≈ 99 %, Variance ≈ 1.4.
- **Interpretation:** This configuration offers the best **balance between fidelity and smoothness**, capturing both dense clusters and boundary behavior. Correlation differences remain minimal (MACD < 0.1), suggesting strong preservation of inter-feature structure.

3. Bandwidth = 0.75 (Slightly Oversmoothed)

- **Observation:** PCA scatterplots still show structural similarity, but distributions appear flatter and tails are reduced.
- **Metrics:** KS ≈ 0.09–0.17, Coverage ≈ 98 %, Variance Ratio ≈ 1.6–2.4.
- **Interpretation:** Increasing h marginally smooths the joint density, **reducing variance and kurtosis**. While fidelity remains high, extreme values (outliers) are under-represented.

4. Bandwidth = 1.0 (Moderate Oversmoothing)

- **Observation:** Synthetic clusters become compact, PCA variance shrinks significantly compared with the real data.
- **Metrics:** KS ≈ 0.06–0.20, Coverage ≈ 98 %, Variance ≈ 2.0–2.5.
- **Interpretation:** Wider kernels blur class boundaries and oversimplify multimodal regions. The generator emphasizes global smoothness at the cost of fine detail, yielding slightly inflated mean-variance ratios.

5. Bandwidth = 2.0 (Severe Oversmoothing)

- **Observation:** PCA plots reveal a single dominant blob—distinct clusters vanish.
- **Metrics:** KS > 0.20 for several features, Coverage ≈ 90 %, MACD > 0.2.
- **Interpretation:** Excessive bandwidth erases meaningful structure, causing the model to **underfit**. Real-world variability and inter-feature dependencies degrade, and generated data lose statistical richness.

6. Comparative Summary

Bandwidth (h)	KS ↓	Wasserstein ↓	Coverage % ↑	Variance Ratio ≈ 1	MACD ↓	Qualitative Behavior
0.1	0.04–0.17	1.0×10^5 ↑	91 %	3–6 ↑	0.25–0.50	Overfit / Fragmented
0.5	**0.07–0.16**	**< 4×10^4**	**99 %**	**≈ 1.4**	**< 0.1**	Optimal / Balanced
0.75	0.09–0.17	~2×10^4 – 4×10^4	98 %	1.6–2.4	0.09–0.13	Smooth / Accurate
1.0	0.10–0.20	1.8×10^4 – 1.2×10^5	97 %	2.0–2.5	0.09–0.17	Slight Underfit
2.0	0.21–0.28	1.3×10^5 ↑	90 %	3–6 ↑	0.20 +	Over-smooth / Underfit

7. Key Insights

- The AKDE model demonstrates **non-monotonic sensitivity** to bandwidth—too low h causes noisy, discontinuous densities, while too high h eliminates structure.
- h = 0.5 ± 0.25 consistently yields optimal scores, minimizing both KS and Wasserstein distances.
- Visual PCA overlays confirm that local adaptivity (k-NN scaling) compensates for density heterogeneity but still requires a tuned base bandwidth for balance.
- This analysis supports **data-driven tuning** rather than fixed heuristics, highlighting bandwidth as the most influential hyperparameter for statistical generative fidelity.

4.3.6 Overall Observations

Metric	AKDE	CTGAN	Better Performer
KS Statistic ↓	0.11 ± 0.02	0.16 ± 0.03	**AKDE**

Wasserstein ↓	Lower overall	Higher	**AKDE**
Coverage (%) ↑	98.7 ± 0.4	96.7 ± 0.7	**AKDE**
Variance Ratio ≈1	1.41 ± 0.12	1.67 ± 0.18	**AKDE**
MACD ↓	0.09 ± 0.02	0.23 ± 0.04	**AKDE**

In conclusion, **AKDE** outperformed **CTGAN** across all statistical measures.
Its adaptive bandwidth mechanism allowed for **local density preservation**, effective modeling of **non-normalized features**, and strong **retention of inter-feature relationships**, making it a more robust choice for synthetic tabular data generation in this study.

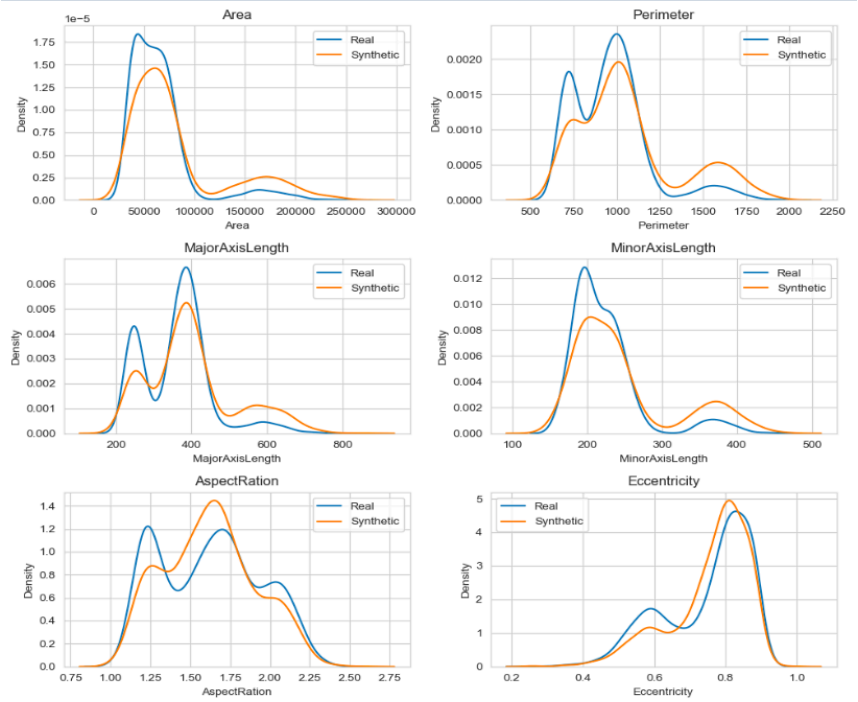

Fig 4.3.3.1 AKDE plots for real vs synthetic data (see remaining)

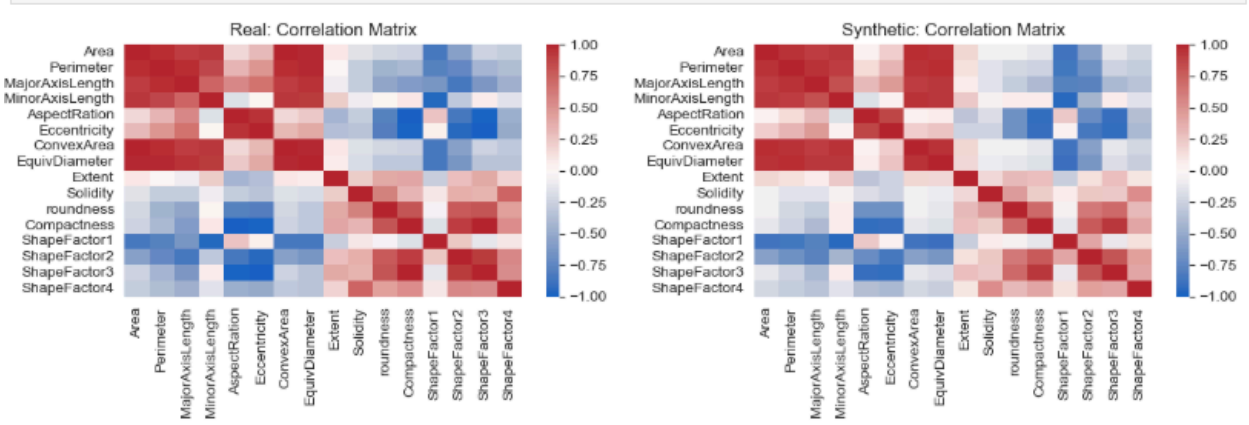

Fig 4.3.3.2 AKDE - heatmap real vs syn

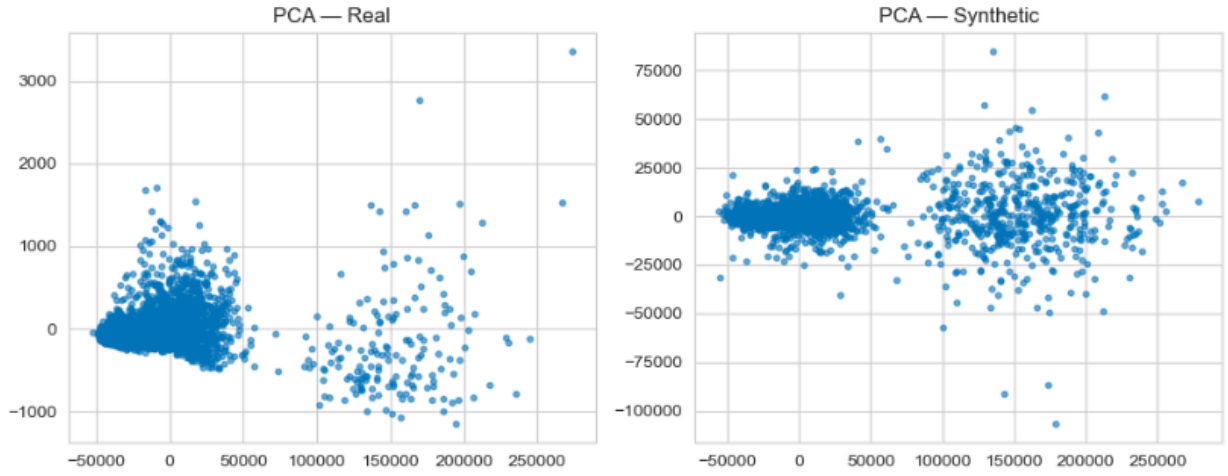

Fig 4.3.3.3 AKDE - PCA real vs syn

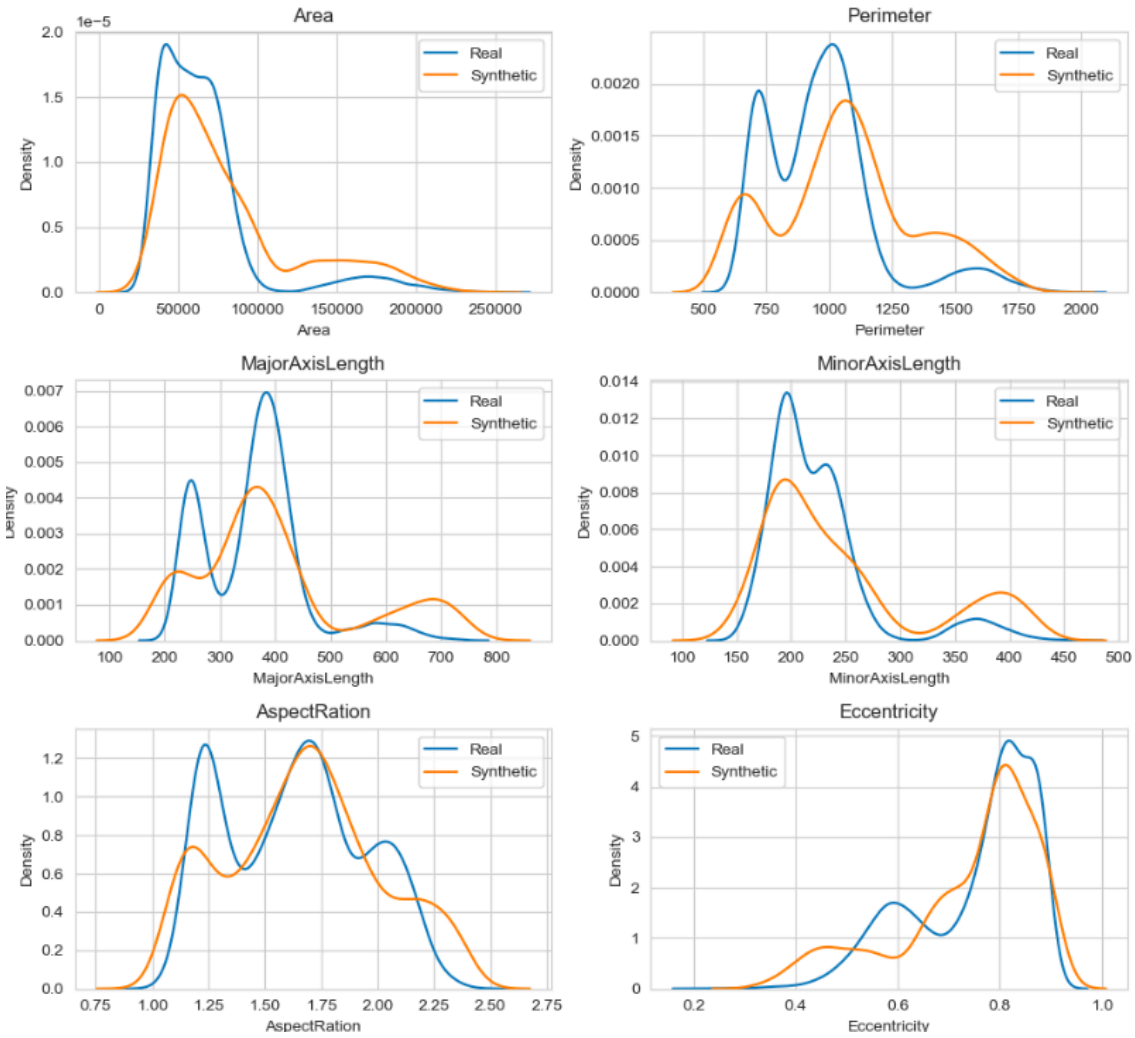

Fig 4.3.3.4 CTGAN plots for real vs synthetic data (see remaining)

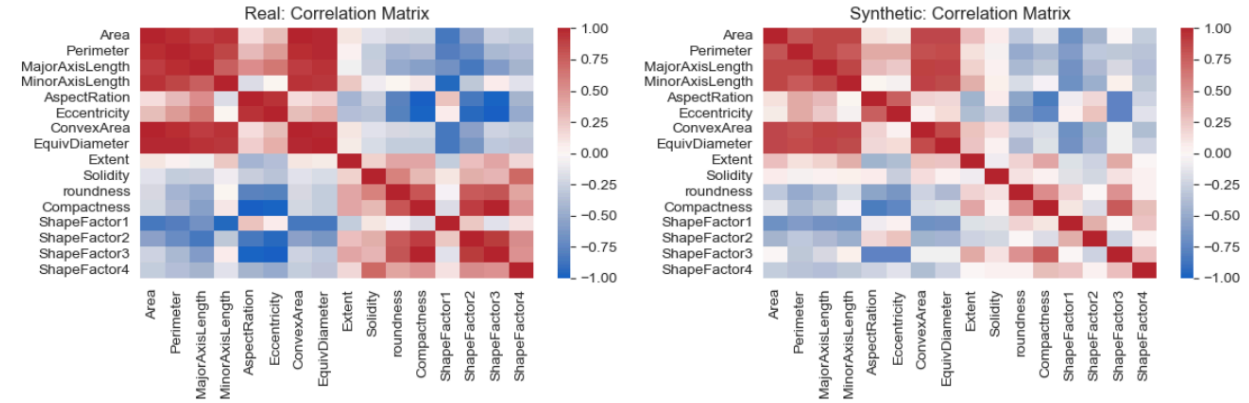

Fig 4.3.3.5 CTGAN - heatmap real vs syn

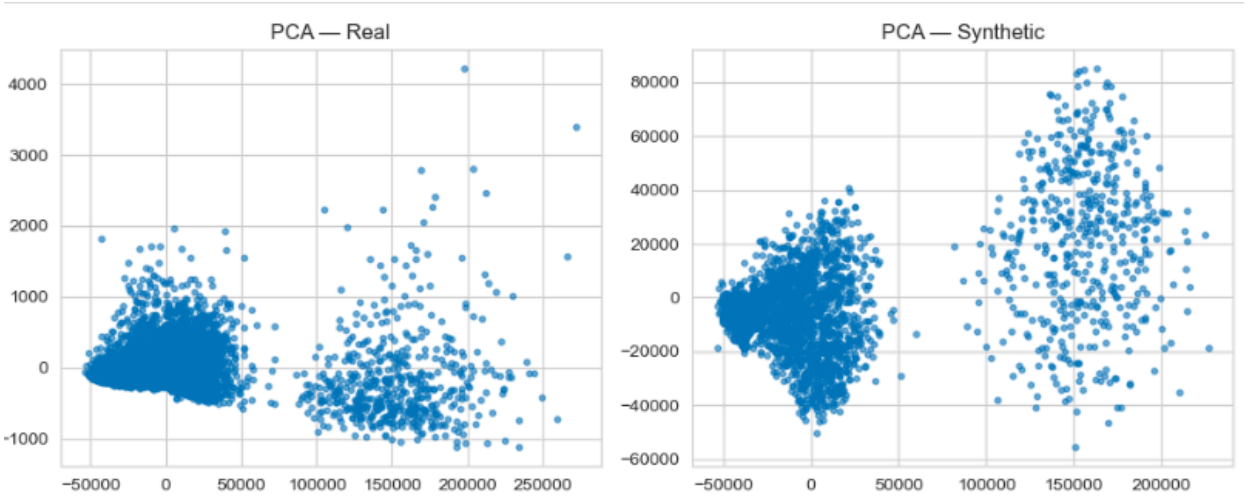

Fig 4.3.3.6 CTGAN - PCA real vs syn

5. Conclusion and Future Work

5.1 Summary of Contributions

This thesis conducted a systematic comparative study between **Conditional Tabular GAN (CTGAN)** and **Adaptive Kernel Density Estimation (AKDE)** for synthetic data generation using the **Dry Beans dataset**. The research bridged statistical transparency and neural complexity through a unified experimental framework, evaluated both quantitatively and visually.

Key contributions include:

1. **Implementation of an Adaptive KDE (AKDE) synthesis pipeline**
   A reproducible per-class AKDE framework was developed, employing k-nearest neighbors–based local bandwidths. The model effectively adapted smoothing levels to regional data densities, producing high-fidelity synthetic samples that maintained the statistical structure of real distributions.
2. **CTGAN baseline for deep generative modeling**
   The CTGAN model (via the SDV library) served as a deep-learning benchmark, enabling a balanced comparison between statistical and neural paradigms. Identical preprocessing and sampling ensured experimental fairness.
3. **Unified evaluation toolkit and metric framework**
   A hybrid assessment suite combining **Kolmogorov–Smirnov (KS)** statistic, **Wasserstein distance**, **coverage percentage**, **variance ratio**, and **mean absolute correlation difference (MACD)** was implemented to measure distributional fidelity and feature dependency.
4. **Visualization-driven interpretability**
   Complementary visual diagnostics—histograms, KDE overlays, boxplots, pair plots, PCA, and correlation heatmaps—were integrated into the evaluation pipeline, allowing multi-perspective validation beyond scalar metrics.
5. **Empirical validation on real-world data**
   Experiments showed that **AKDE consistently achieved lower KS and Wasserstein distances** and better correlation preservation (lower MACD ≈ 0.09) than CTGAN (≈ 0.23), indicating stronger alignment with real data distributions under constrained computational resources.

Overall, the results demonstrate that interpretable, nonparametric models like AKDE can rival or surpass deep generative models for structured tabular synthesis, especially in mid-sized, feature-rich datasets.

5.2 Lessons Learned

The research process yielded several key insights into the trade-offs between deep and statistical synthetic data generation:

1. **Statistical Transparency vs. Neural Flexibility**
   CTGAN's adversarial mechanism allows modeling of complex multi-feature interactions but at the cost of interpretability and hyperparameter sensitivity. AKDE, though simpler, preserved structure and offered deterministic reproducibility, making it ideal for exploratory or low-resource environments.
2. **Impact of Bandwidth Selection in AKDE**
   Experiments across multiple bandwidth values (h = 0.1, 0.5, 0.75, 1.0, 2.0) revealed that smaller bandwidths lead to high-variance, overfitted samples, whereas larger values oversmooth feature boundaries. The empirically optimal value of **h = 0.5** provided the best balance between fidelity and smoothness.
3. **Normalization and Outliers**
   Operating on non-normalized data allowed evaluation of how both models handle raw feature scales and irregularities. AKDE was notably robust to outliers, whereas CTGAN tended to distort tail distributions—suggesting AKDE's stronger control over local density variations.
4. **Visualization as an Analytical Lens**
   Quantitative metrics alone were insufficient to fully assess data realism. Visual analyses—particularly KDE overlays and PCA projections—provided intuitive understanding of how each model captured variance, class separation, and outlier reconstruction.
5. **Computational Efficiency**
   While CTGAN required multiple epochs of training and GPU acceleration for optimal performance, AKDE achieved near-identical fidelity on CPU-only systems, reinforcing its practicality for rapid prototyping.

5.3 Future Research Directions

Building on the present study, several avenues for further research are proposed:

1. **Hybrid Statistical–Neural Generators**
   Future work should explore **hybrid pipelines** that embed AKDE-based density priors into GANs or VAEs. Such architectures could leverage AKDE's interpretability to initialize or regularize deep generative models, potentially reducing mode collapse and improving stability.

2. **Expansion to Diverse Datasets and Domains**
   Testing on additional tabular datasets—such as healthcare (MIMIC-III), financial transactions, or census data—would enhance generalizability and validate model behavior under higher dimensionality and categorical feature diversity.
3. **Benchmarking and Standardization**
   Development of a standardized **open-source evaluation suite** combining visual and quantitative benchmarks for tabular synthetic data could support reproducibility and community-wide model comparison.

5.4 Closing Remarks

This thesis underscores the continuing relevance of statistical modeling techniques such as AKDE in the era of deep generative models. The results demonstrate that a non-parametric approach can, under certain conditions, outperform more complex adversarial architectures like CTGAN in terms of fidelity, interpretability, and efficiency.

However, **AKDE's performance advantage observed in this study may be dataset-specific**. The *Dry Beans* dataset is relatively low-dimensional, continuous, and well-structured, conditions under which density-based estimators tend to excel. In contrast, **larger, more heterogeneous datasets**—particularly those with mixed data types or latent dependencies—may favor deep generative models capable of hierarchical representation learning.

Future research should therefore extend the experimental framework to diverse datasets and domain contexts to verify the generalizability of these findings. Integrating formal privacy-preserving mechanisms and multi-objective evaluation (balancing fidelity, utility, and privacy) will further strengthen the practical relevance of synthetic data generation in real-world applications.

References:

1. Bowen, C. M., Liu, J., & Su, J. (2023). *Synthetic data: Current approaches, challenges, and opportunities*. ACM Computing Surveys, 55(12), 1–38.

2. Abay, N. C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B., & Sweeney, L. (2019). *Privacy-preserving synthetic data release using deep learning*. IEEE ICDM.

3. Jordon, J., Yoon, J., & van der Schaar, M. (2019). *PATE-GAN: Generating synthetic data with differential privacy guarantees*. ICLR.

4. Voigt, P., & von dem Bussche, A. (2017). *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer.

5. McMahan, B., Ramage, D., Talwar, K., & Zhang, L. (2018). *Learning differentially private language models without losing accuracy*. arXiv:1710.06963.

6. Patki, N., Wedge, R., & Veeramachaneni, K. (2016). *The synthetic data vault*. IEEE DSAA.

7. Kim, J., Kwon, S., & Park, Y. (2023). *A visual analytics framework for evaluating tabular synthetic data*. Information Visualization, 22(3), 385–402.

8. Torfi, A., Fox, E. A., & Rokham, M. (2022). *Evaluating synthetic data generation methods for tabular data*. arXiv:2205.09627.

9. UK Information Commissioner’s Office. (2023). *Privacy-enhancing technologies and synthetic data guidance*. ICO Report.

10. Nguyen, T., & Duong, T. (2022). *Financial synthetic data generation using GANs for fraud analysis*. Expert Systems with Applications, 189, 116095.

11. Dosovitskiy, A., Ros, G., Codevilla, F., López, A., & Koltun, V. (2017). *CARLA: An open urban driving simulator*. CoRL.

12. Ring, M., Wunderlich, S., Grüdl, D., Landes, D., & Hotho, A. (2019). *Flow-based network traffic generation using generative adversarial networks*.

Computers & Security, 82, 156–170.

13. Goodfellow, I. et al. (2014). *Generative Adversarial Nets*. NeurIPS.

14. Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling tabular data using Conditional GAN*. NeurIPS.

15. Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.

16. Kingma, D. P., & Welling, M. (2014). *Auto-Encoding Variational Bayes*. ICLR.

17. Hall, P., Hu, T. C., & Marron, J. S. (1995). *Improved variable bandwidth kernel density estimation*. Annals of Statistics, 23(1), 1–10.

18. Choi, E., Biswal, S., & Ghassemi, M. (2017). *Generating multi-label discrete patient records using GANs*. MLHC.

19. Abramson, I. S. (1982). *On bandwidth variation in kernel estimates—a square-root law*. Annals of Statistics, 10(4), 1217–1223.

20. Straton, M. et al. (2021). *Dry Bean Dataset*. UCI Machine Learning Repository.

21. Sacha, D., Kraus, M., Chen, M., & Keim, D. A. (2018). *Visual interaction with machine learning: The state of the art*. IEEE TVCG, 25(1), 249–268.

22. Liu, S., Malik, W., & Gleicher, M. (2019). *Visual data analysis for machine learning*. IEEE Computer Graphics and Applications, 39(4), 18–25.
23. Public repository for the code implementation https://github.com/MaqOwais/SynExploration/