CSCA 5642: Introduction to Deep Learning

University of Colorado Boulder

Synthetic Brand Generation with Ensemble Methods

Dyego Fernandes de Sousa

University of Colorado Boulder

December 6, 2025

Problem Statement and Data Overview

To showcase the application of **Generative Deep Learning** techniques for synthesizing realistic data.
This project is a continuation of my previous works on **Supervised Learning** and **Unsupervised Learning**, you can find more information in the appendix.

Detailed ESG (Environmental Social and Governance) dataset to the brand-level

Observations:	**3605**
Features:	**77**

ML Approach & Architecture

Model	Type	Purpose	Key Features
CTGAN[1]	Conditional GAN	Tabular synthesis	Mode-specific normalization, conditional vector
TVAE[2]	Variational Autoencoder	Distribution learning	KL divergence, latent space regularization
Gaussian Copula	Statistical	Correlation preservation	Multivariate dependencies
GPT-2 Medium	Transformer LLM	Brand name generation	355M params, fine-tuned
Flan-T5 Small	Encoder-Decoder	Conditional text gen	Instruction-following

Ensemble Methods: Combine multiple generators with optimized weighting
Evaluation Metrics: Apply statistical tests (KS, correlation) to assess synthetic data quality

[1] Conditional Tabular Generative Adversarial Networks [2] Tabular Variational Autoencoders

ML Approach Architecture

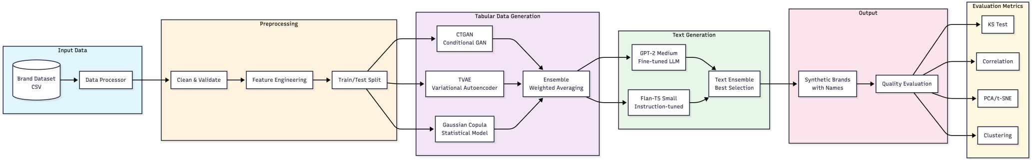


Figure: Pipeline

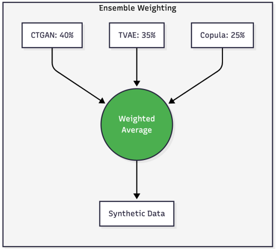


Figure: Ensemble Architecture

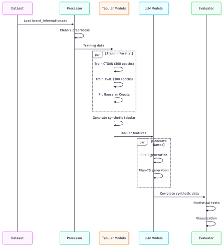


Figure: Sequence Diagram

Data Preprocessing and Exploration

Performed Data Cleaning, Handling Missing Values, Encoding Categorical Variables, and Feature Engineering.

```
# Load and process data
processor = BrandDataProcessor(DATA_PATH)
raw_data = processor.load_data()
```

```
# Prepare for GAN training
train_df, val_df = processor.prepare_for_gan(test_size=0.2)

discrete_cols = processor.categorical_features
binary_cols = [col for col in train_df.columns if train_df[col].nunique() == 2
    and set(train_df[col].unique()).issubset({0, 1})]
numerical_cols = [col for col in train_df.columns if col not in discrete_cols
    and col not in binary_cols]
```

```
# Clean data
cleaned_data = processor.clean_data()
```

Hyperparameter Tuning

Using Optuna to find optimal hyperparameters for the tabular synthesizers.

```python
# Create tuner instance
tuner = HyperparameterTunerV2(
    train_data=train_df,
    discrete_cols=discrete_cols,
    binary_cols=binary_cols,
    eval_sample_size=min(1000, len(train_df)),
    gen_sample_size=500,
    verbose=True
)

# Run optimization
best_hyperparams = tuner.tune(
    n_trials=N_TUNING_TRIALS,
    timeout=TUNING_TIMEOUT,
    seed=42,
    show_progress_bar=True
)

# Save the best hyperparameters
tuner.save(HYPERPARAMS_PATH)
```

Best Parameters:

Parameter	Value
ctgan_epochs	500
tvae_epochs	400
batch_size	500
embedding_dim	256
generator_dim	[256, 256]
discriminator_dim	[128, 128]

Training CTGAN, TVAE, and Gaussian Copula models

```python
# Initialize Ensemble Synthesizer
tabular_ensemble = EnsembleSynthesizer(
    ctgan_epochs=CTGAN_EPOCHS,
    ctgan_batch_size=BATCH_SIZE,
    tvae_epochs=TVAE_EPOCHS,
    tvae_batch_size=BATCH_SIZE,
    gc_default_distribution='beta',
    weights=ENSEMBLE_WEIGHTS,
    verbose=True,
    cuda=True
)
```

```python
training_times = tabular_ensemble.train(
    data=train_df,
    discrete_columns=discrete_cols,
    binary_columns=binary_cols
)
# Save models
tabular_ensemble.save_models(os.path.join(MODEL_DIR, 'tabular_ensemble'))
# Optimize weights based on quality
optimized_weights = tabular_ensemble.optimize_weights(train_df, n_eval_samples=1000)
```

LLM Ensemble Training

Training GPT-2 Medium and Flan-T5 for brand name generation

```python
# Initialize LLM Ensemble Generator
llm_generator = BrandNameGeneratorV2(
    models=LLM_MODELS,
    memory_efficient=True,
    verbose=True
)
```

```python
llm_generator.fine_tune(
    brands_df=brands_df,
    epochs=LLM_EPOCHS,
    output_dir=os.path.join(MODEL_DIR, '
        llm_ensemble')
)
# Save ensemble config
llm_generator.save_model(os.path.join(MODEL_DIR, '
    llm_ensemble'))
```

Figure: Fine Tuning LLMs

Company Name	1st	2nd	3rd
PepsiCo	Coca Cola	Dr Pepper	Snapple's
Nestle	Tomatoes	Nutella	Nestle is a manufacturer...
Mars, Incorporated	Whole30	Kashi	Soylent

Table: Generated Brand Names by Model

Synthetic Data Generation

```python
# Generate synthetic tabular features using ensemble
print("Generating synthetic features with ensemble...")
synthetic_features, failed_companies = tabular_ensemble.generate_stratified(
    company_distribution=generation_targets,
    verbose=True
)
```

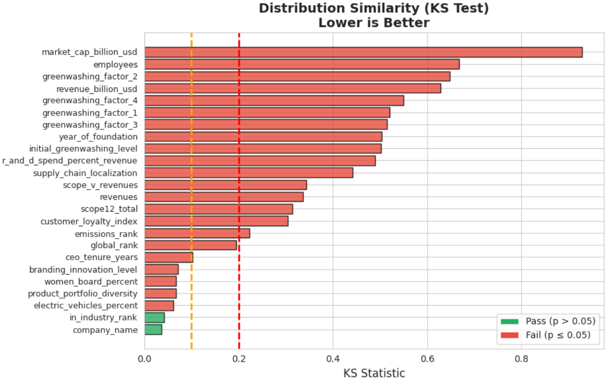
```python
# Generate brand names using LLM ensemble
synthetic_with_names = llm_generator.generate_for_dataframe(
    synthetic_df=synthetic_decoded,
    temperature=DIVERSITY_TEMPERATURE,
    verbose=True
)
```

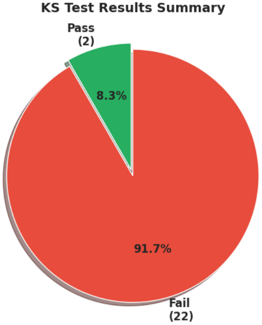
Figure: Generating Synthetic Tabular Data and Brand Names

Synthetic Data Quality Evaluation

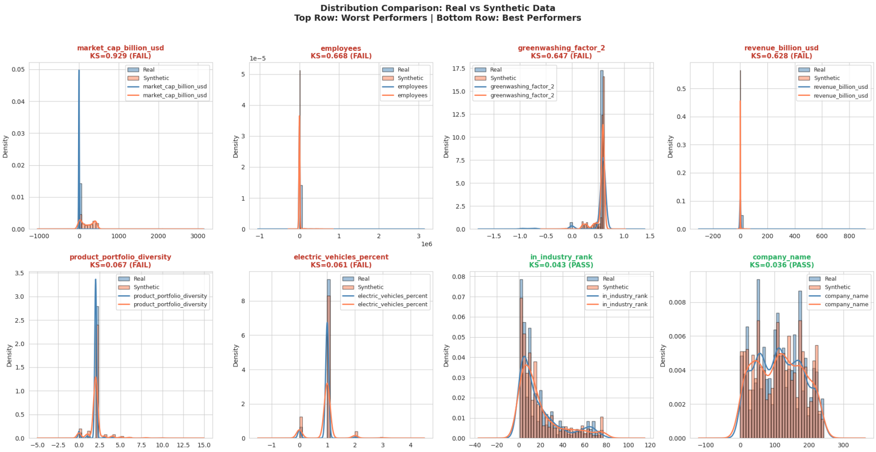
This phase comprehensively evaluates the quality of generated synthetic data through:

- **Tabular Data**: Prioritizing variance over fidelity for data augmentation.
- **Dimensionality Reduction**: PCA and t-SNE projections

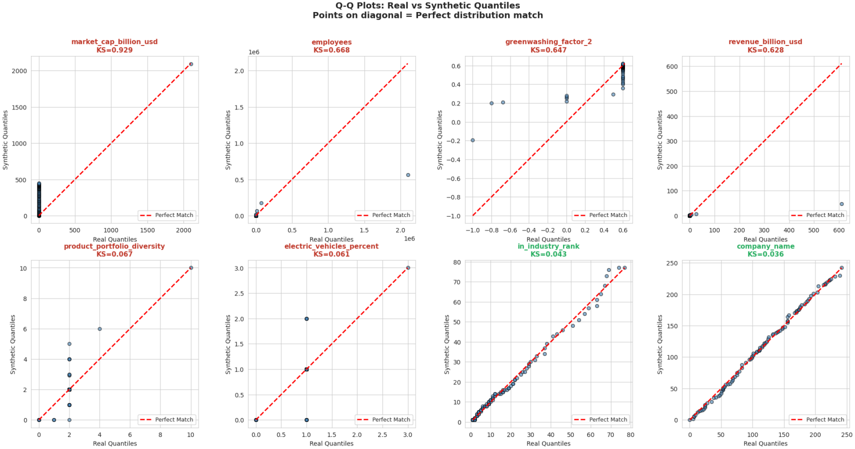




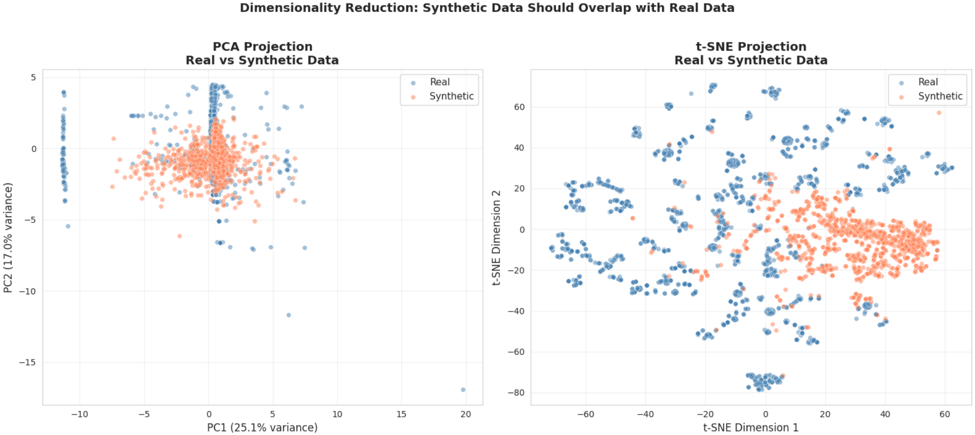
Distribution Comparison



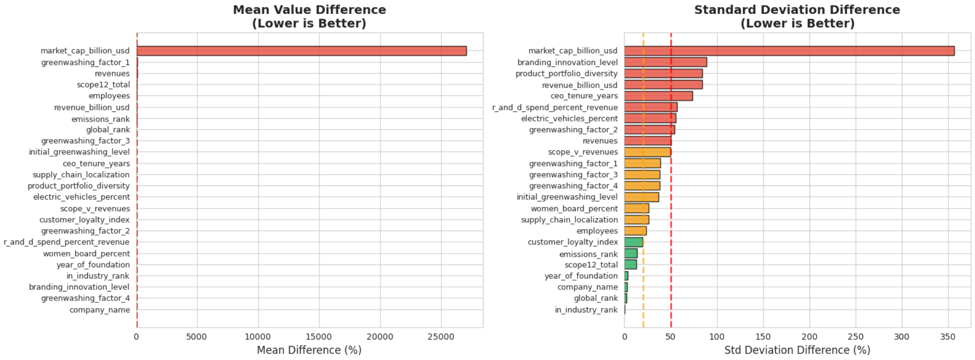
QQ Plot Comparison



Dimensionality Reduction



Statistical Comparison



Conclusions, summary and future work

- Successful implementation of an ensemble-based synthetic data generation pipeline combining tabular synthesizers (CTGAN, TVAE, Gaussian Copula) and LLMs (GPT-2, Flan-T5) to create realistic brand-level ESG datasets.
- Trade-offs between fidelity and variance were effectively managed to produce diverse yet representative synthetic data.
- Optimal models in production-ready state with scalable architecture for future enhancements.

		Details
Worked Well	Ensemble Architecture	CTGAN, TVAE, GC complemented; TVAE 54% weight (KS: 0.11), GC preserved correlations (MSE: 0.0197)
	Scalable Pipeline	Stratified generation, conditional synthesis, Google Drive persistence
	LLM Brand Names	GPT-2 + Flan-T5 achieved 95.4% success rate, memory-efficient
Challenges	LLM Quality Issues	Full sentences, competitor leakage, repetitive patterns, 4.6% fallback
Future Work	Tabular Synthesis	Feature preprocessing, constrained generation, TabDDPM, validation
	LLM Enhancement	Negative examples, stricter validation, RAG, style conditioning
	Architecture	Attention-based models, hierarchical pipeline, discriminator filtering

References Bibliography

1. **CTGAN (Conditional Tabular GAN)**
   - Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling Tabular Data using Conditional GAN*. NeurIPS 2019.
   - Paper: https://arxiv.org/abs/1907.00503
   - Implementation: SDV Library
2. **TVAE (Tabular Variational Autoencoder)**
   - Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling Tabular Data using Conditional GAN*. NeurIPS 2019.
   - Part of the same paper as CTGAN, presenting VAE-based alternative
3. **Gaussian Copula**
   - Patki, N., Wedge, R., & Veeramachaneni, K. (2016). *The Synthetic Data Vault*. IEEE DSAA 2016.
   - Paper: https://dai.lids.mit.edu/wp-content/uploads/2018/03/SDV.pdf
4. **SDV (Synthetic Data Vault) Library**
   - Documentation: https://docs.sdv.dev/sdv/
   - GitHub: https://github.com/sdv-dev/SDV
5. **GPT-2**
   - Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. OpenAI.
   - Paper: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
6. **Flan-T5**
   - Chung, H. W., et al. (2022). *Scaling Instruction-Finetuned Language Models*. arXiv preprint.
   - Paper: https://arxiv.org/abs/2210.11416
7. **Hugging Face Transformers**
   - Wolf, T., et al. (2020). *Transformers: State-of-the-Art Natural Language Processing*. EMNLP 2020.
   - Documentation: https://huggingface.co/docs/transformers/
8. **Kolmogorov-Smirnov Test**
   - Massey Jr, F. J. (1951). *The Kolmogorov-Smirnov test for goodness of fit*. Journal of the American Statistical Association, 46(253), 68-78.
9. **Silhouette Score**
   - Rousseeuw, P. J. (1987). *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*. Journal of Computational and Applied Mathematics, 20, 53-65.
10. **Davies-Bouldin Index**
    - Davies, D. L., & Bouldin, D. W. (1979). *A cluster separation measure*. IEEE Transactions on Pattern Analysis and Machine Intelligence, (2), 224-227.
11. **Optuna**
    - Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. KDD 2019.
    - Paper: https://arxiv.org/abs/1907.10902
    - Documentation: https://optuna.org/
12. **TabDDPM (Diffusion Models for Tabular Data)**
    - Kotelnikov, A., Baranchuk, D., Rubachev, I., & Babenko, A. (2023). *TabDDPM: Modelling Tabular Data with Diffusion Models*. ICML 2023.
    - Paper: https://arxiv.org/abs/2209.15421
13. **CTAB-GAN+**
    - Zhao, Z., Kunar, A., Birke, R., & Chen, L. Y. (2022). *CTAB-GAN+: Enhancing Tabular Data Synthesis*. arXiv preprint.
    - Paper: https://arxiv.org/abs/2204.00401
14. **Synthetic Data Generation Survey**
    - Jordon, J., Yoon, J., & van der Schaar, M. (2022). *Synthetic Data - what, why and how?* arXiv preprint.
    - Paper: https://arxiv.org/abs/2205.03257

Appendix Other Repositories

- **Supervised Learning**: https://github.com/dyegofern/csca5622-supervised-learning
- **Unsupervised Learning**: https://github.com/dyegofern/csca5632-unsupervised-learning

Appendix Mermeid Code

```
flowchart TB
    subgraph INPUT["Input Data"]
        A[(Brand Dataset<br/>CSV)] --> B[Data Processor]
    end
    subgraph PREPROCESS["Preprocessing"]
        B --> C[Clean \& Validate]
        C --> D[Feature Engineering]
        D --> E[Train/Test Split]
    end
    subgraph TABULAR["Tabular Data Generation"]
        E --> F1[CTGAN<br/>Conditional GAN]
        E --> F2[TVAE<br/>Variational Autoencoder]
        E --> F3[Gaussian Copula<br/>Statistical Model]
        F1 --> G[Ensemble<br/>Weighted Averaging]
        F2 --> G
        F3 --> G
    end
    subgraph TEXT["Text Generation"]
        G --> H1[GPT-2 Medium<br/>Fine-tuned LLM]
        G --> H2[Flan-T5 Small<br/>Instruction-tuned]
        H1 --> I[Text Ensemble<br/>Best Selection]
        H2 --> I
    end
    subgraph OUTPUT["Output"]
        I --> J[Synthetic Brands<br/>with Names]
        J --> K[Quality Evaluation]
    end
    subgraph EVAL["Evaluation Metrics"]
        K --> L1[KS Test]
        K --> L2[Correlation]
        K --> L3[PCA/t-SNE]
        K --> L4[Clustering]
    end
    style INPUT fill:\#e1f5fe
    style PREPROCESS fill:\#fff3e0
    style TABULAR fill:\#f3e5f5
    style TEXT fill:\#e8f5e9
    style OUTPUT fill:\#fce4ec
    style EVAL fill:\#fff8e1
```

```
flowchart LR
    subgraph ENSEMBLE["Ensemble Weighting"]
        direction TB
        W1["CTGAN: 40\%"] --> MIX((Weighted<br/>Average))
        W2["TVAE: 35\%"] --> MIX
        W3["Copula: 25\%"] --> MIX
        MIX --> OUT[Synthetic Data]
    end
    style ENSEMBLE fill:\#f5f5f5
    style MIX fill:\#4caf50,color:\#fff
```

```
sequenceDiagram
    participant D as Dataset
    participant P as Processor
    participant T as Tabular Models
    participant L as LLM Models
    participant E as Evaluator
    D->>P: Load brand\_information.csv
    P->>P: Clean \& preprocess
    P->>T: Training data
    par Train in Parallel
        T->>T: Train CTGAN (300 epochs)
        T->>T: Train TVAE (300 epochs)
        T->>T: Fit Gaussian Copula
    end
    T->>T: Generate synthetic tabular
    T->>L: Tabular features
    par Generate Names
        L->>L: GPT-2 generation
        L->>L: Flan-T5 generation
    end
    L->>E: Complete synthetic data
    E->>E: Statistical tests
    E->>E: Visualization
```

Appendix Tools & Libraries

- **PyTorch** - Deep learning framework powering the neural network components of CTGAN and TVAE synthesizers
- **Scikit-learn** - Machine learning utilities for PCA, t-SNE, clustering (AgglomerativeClustering), and evaluation metrics (silhouette score, Davies-Bouldin index)

- **SDV (Synthetic Data Vault)** - Primary library for tabular synthetic data generation, providing:
- **CTGAN** - Conditional Tabular GAN for generating realistic tabular data
- **TVAE** - Tabular Variational Autoencoder for distribution-preserving synthesis
- **Gaussian Copula** - Statistical model capturing feature dependencies

- **Hugging Face Transformers** - State-of-the-art NLP library for text generation using pre-trained language models
- **PEFT (Parameter-Efficient Fine-Tuning)** - Efficient fine-tuning techniques for large language models
- **BitsAndBytes** - 8-bit quantization for memory-efficient model loading
- **Accelerate** - Distributed training and mixed precision utilities
- **SentencePiece** - Tokenization library for handling text preprocessing

- **Optuna** - Automated hyperparameter tuning framework using Bayesian optimization for finding optimal model configurations
- **Pandas** - Data manipulation and analysis
- **NumPy** - Numerical computing and array operations
- **SciPy** - Statistical tests (Kolmogorov-Smirnov test) for distribution comparison

- **Matplotlib** - Core plotting library for creating figures
- **Seaborn** - Statistical data visualization with enhanced aesthetics
- **Plotly** - Interactive visualizations (if used)

- **Google Colab** - Cloud-based Jupyter notebook environment with GPU support
- **Google Drive** - Persistent storage for models and outputs