

# 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 <sup>1</sup>	Conditional GAN	Tabular synthesis	Mode-specific normalization, conditional vector
TVAE <sup>2</sup>	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

---

<sup>1</sup> Conditional Tabular Generative Adversarial Networks    <sup>2</sup> Tabular Variational Autoencoders

# ML Approach Architecture

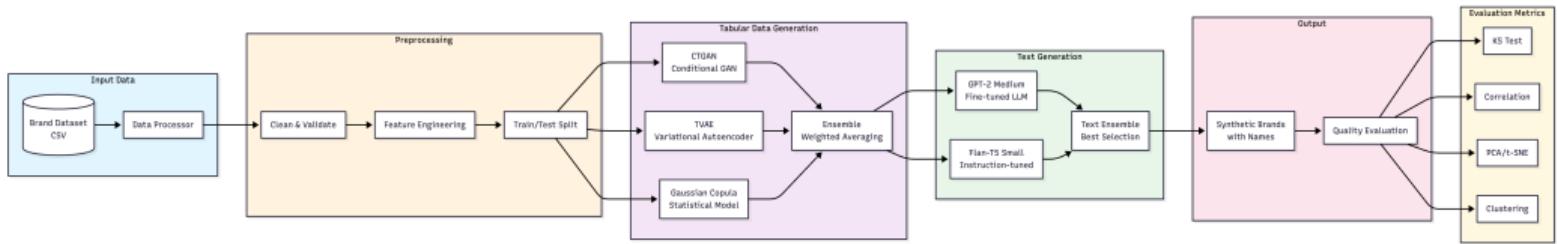


Figure: Pipeline

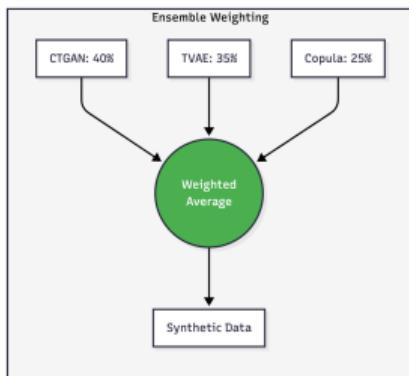


Figure: Ensemble Architecture

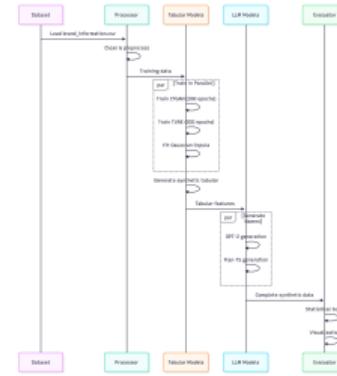


Figure: Sequence Diagram

# Data Preprocessing and Exploration

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

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

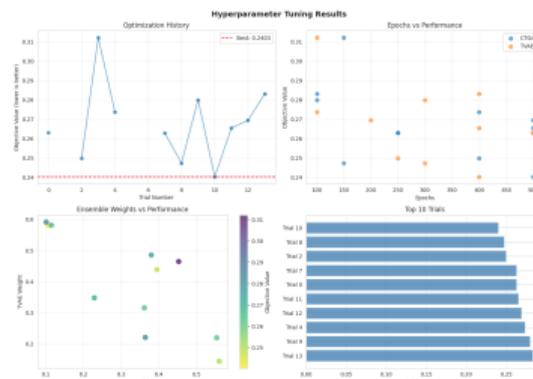
```
1 # Prepare for GAN training
2 train_df, val_df = processor.prepare_for_gan(test_size=0.2)
3
4 discrete_cols = processor.categorical_features
5 binary_cols = [col for col in train_df.columns if train_df[col].nunique() == 2
    and set(train_df[col].unique()).issubset({0, 1})]
6 numerical_cols = [col for col in train_df.columns if col not in discrete_cols
    and col not in binary_cols]
```

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

# Hyperparameter Tuning

Using Optuna to find optimal hyperparameters for the tabular synthesizers.

```
1 # Create tuner instance
2 tuner = HyperparameterTunerV2(
3     train_data=train_df,
4     discrete_cols=discrete_cols,
5     binary_cols=binary_cols,
6     eval_sample_size=min(1000, len(train_df)),
7     gen_sample_size=500,
8     verbose=True
9 )
10
11 # Run optimization
12 best_hyperparams = tuner.tune(
13     n_trials=N_TUNING_TRIALS,
14     timeout=TUNING_TIMEOUT,
15     seed=42,
16     show_progress_bar=True
17 )
18
19 # Save the best hyperparameters
20 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 Tabular Ensemble Models

## Training CTGAN, TVAE, and Gaussian Copula models

```
1 # Initialize Ensemble Synthesizer
2 tabular_ensemble =
3     EnsembleSynthesizer(
4         ctgan_epochs=CTGAN_EPOCHS,
5         ctgan_batch_size=BATCH_SIZE,
6         tvae_epochs=TVAE_EPOCHS,
7         tvae_batch_size=BATCH_SIZE,
8         gc_default_distribution='beta',
9         weights=ENSEMBLE_WEIGHTS,
10        verbose=True,
11        cuda=True
12    )
```

```
1 training_times = tabular_ensemble.
2     train(
3         data=train_df,
4         discrete_columns=discrete_cols,
5         binary_columns=binary_cols
6     )
7 # Save models
8 tabular_ensemble.save_models(os.path
9     .join(MODEL_DIR, ,
10       tabular_ensemble'))
11 # Optimize weights based on quality
12 optimized_weights = tabular_ensemble
13     .optimize_weights(train_df,
14     n_eval_samples=1000)
```

# LLM Ensemble Training

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

```
1 # Initialize LLM Ensemble Generator
2 llm_generator = BrandNameGeneratorV2(
3     models=LLM_MODELS,
4     memory_efficient=True,
5     verbose=True
6 )
```

```
1 llm_generator.fine_tune(
2     brands_df=brands_df,
3     epochs=LLM_EPOCHS,
4     output_dir=os.path.join(MODEL_DIR, ,
5                             'llm_ensemble')
6 )
7 # Save ensemble config
8 llm_generator.save_model(os.path.join(MODEL_DIR, ,
9                         '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

```
1 # Generate synthetic tabular features using ensemble
2 print("Generating synthetic features with ensemble...")
3 synthetic_features, failed_companies = tabular_ensemble.generate_stratified(
4     company_distribution=generation_targets,
5     verbose=True
6 )
```

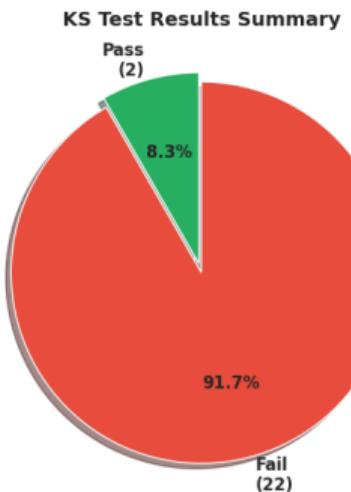
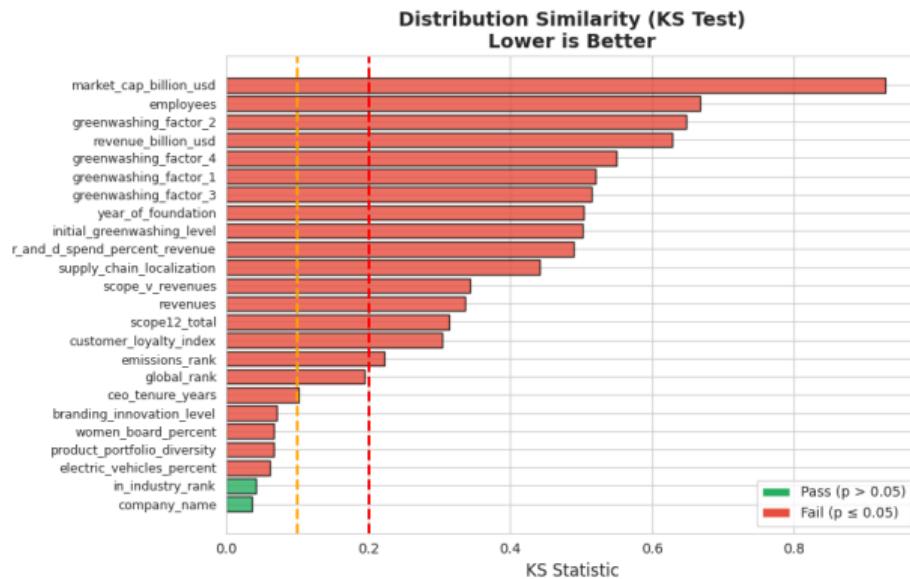
```
1 # Generate brand names using LLM ensemble
2 synthetic_with_names = llm_generator.generate_for_dataframe(
3     synthetic_df=synthetic_decoded,
4     temperature=DIVERSITY_TEMPERATURE,
5     verbose=True
6 )
```

Figure: Generating Synthetic Tabular Data and Brand Names

# Synthetic Data Quality Evaluation

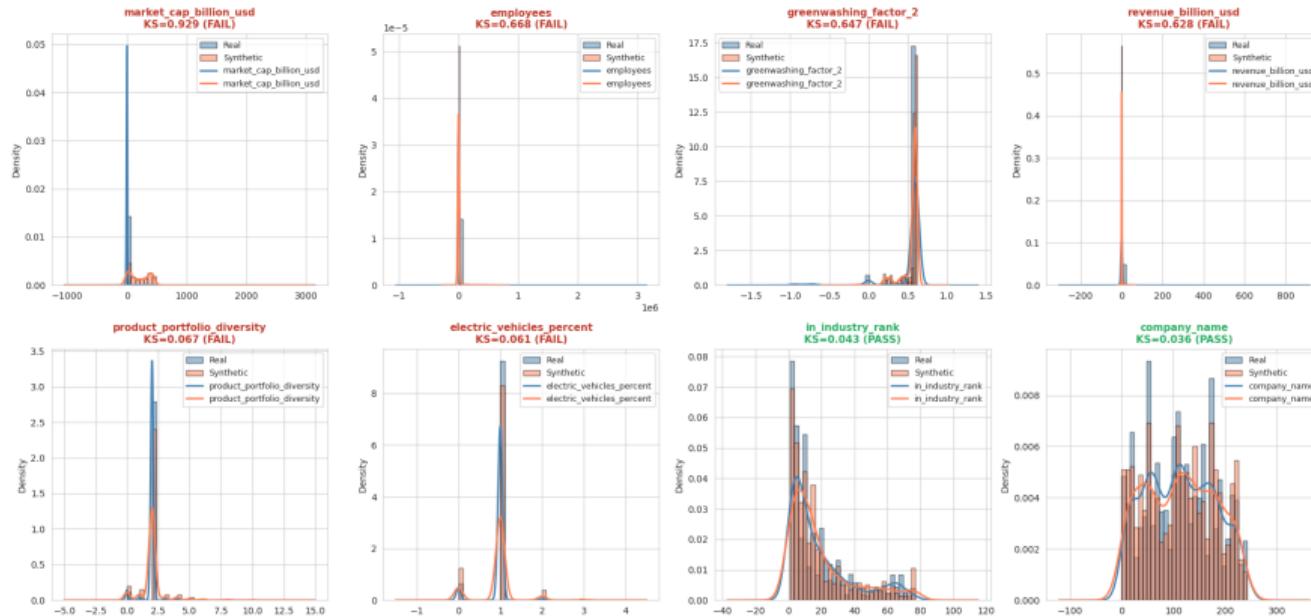
This phase comprehensively evaluates the quality of generated synthetic data through:

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



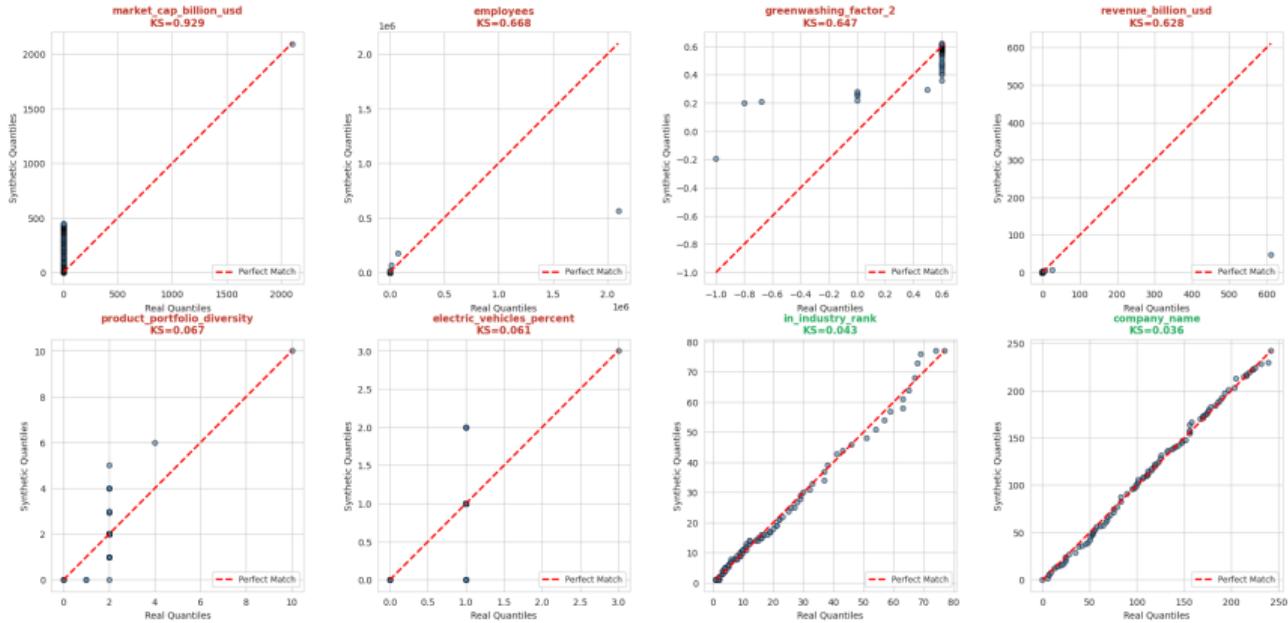
# Distribution Comparison

**Distribution Comparison: Real vs Synthetic Data**  
Top Row: Worst Performers | Bottom Row: Best Performers



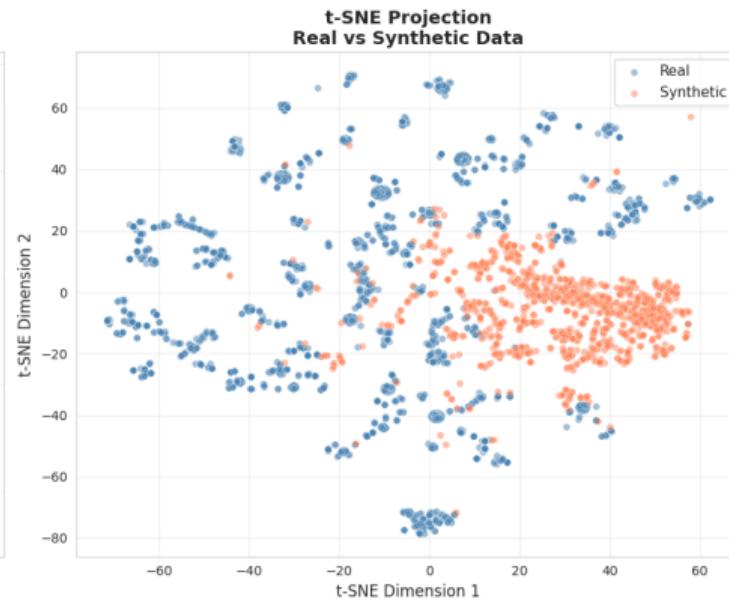
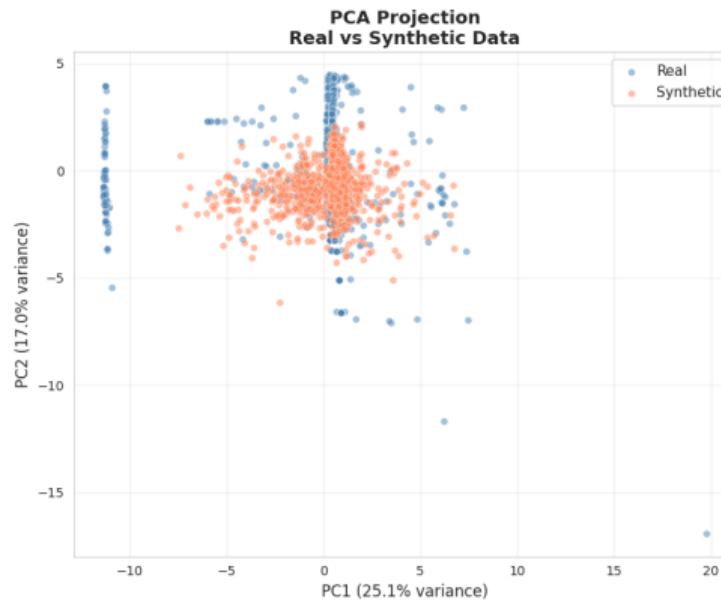
# QQ Plot Comparison

Q-Q Plots: Real vs Synthetic Quantiles  
Points on diagonal = Perfect distribution match

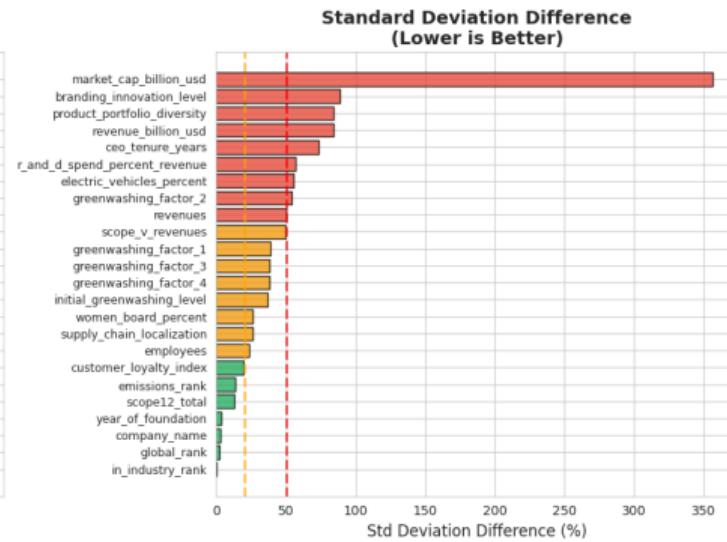
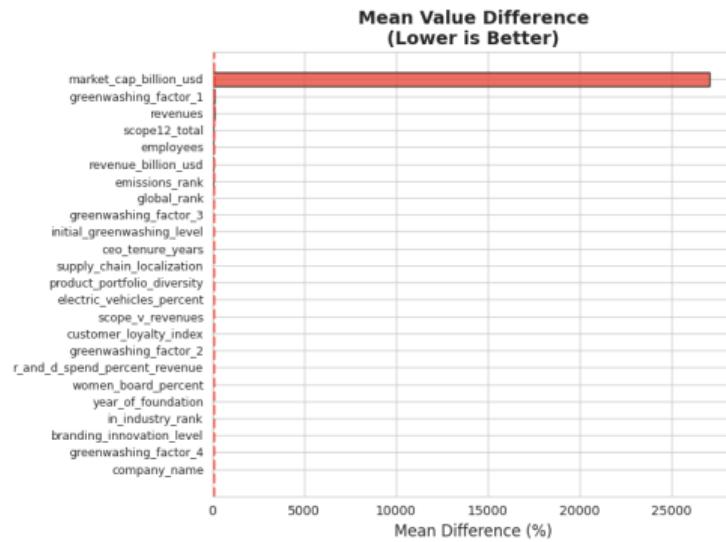


# Dimensionality Reduction

Dimensionality Reduction: Synthetic Data Should Overlap with Real Data



# 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., Skoulikidou, 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., Skoulikidou, 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](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

```
1 flowchart TB
2    subgraph INPUT["Input Data"]
3        A[("Brand Dataset<br/>CSV")] --> B[Data Processor]
4    end
5    subgraph PREPROCESS["Preprocessing"]
6        B --> C[Clean & Validate]
7        C --> D[Feature Engineering]
8        D --> E[Train/Test Split]
9    end
10   subgraph TABULAR["Tabular Data Generation"]
11      E --> F1[CTGAN<br/>Conditional GAN]
12      E --> F2[TVAE<br/>Variational Autoencoder]
13      E --> F3[Gaussian Copula<br/>Statistical Model]
14      F1 --> G[Ensemble<br/>Weighted Averaging]
15      F2 --> G
16      F3 --> G
17  end
18  subgraph TEXT["Text Generation"]
19      G --> H1[GPT-2 Medium<br/>Fine-tuned LLM]
20      G --> H2[Flan-T5 Small<br/>Instruction-tuned]
21      H1 --> I[Text Ensemble<br/>Best Selection]
22      H2 --> I
23  end
24  subgraph OUTPUT["Output"]
25      I --> J[Synthetic Brands<br/>with Names]
26      J --> K[Quality Evaluation]
27  end
28  subgraph EVAL["Evaluation Metrics"]
29      K --> L1[KS Test]
30      K --> L2[Correlation]
31      K --> L3[PCA/t-SNE]
32      K --> L4[Clustering]
33  end
34  style INPUT fill:\#e1f5fe
35  style PREPROCESS fill:\#ffff3e0
36  style TABULAR fill:\#f3e5f5
37  style TEXT fill:\#e8f5e9
38  style OUTPUT fill:\#fce4ec
39  style EVAL fill:\#fff8e1
```

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

```
1 sequenceDiagram
2     participant D as Dataset
3     participant P as Processor
4     participant T as Tabular Models
5     participant L as LLM Models
6     participant E as Evaluator
7     D->>P: Load brand\information.csv
8     P->>P: Clean & preprocess
9     P->>T: Training data
10    par Train in Parallel
11       T->>T: Train CTGAN (300 epochs)
12       T->>T: Train TVAE (300 epochs)
13       T->>T: Fit Gaussian Copula
14    end
15    T->>T: Generate synthetic tabular
16    T->>L: Tabular features
17    par Generate Names
18       L->>L: GPT-2 generation
19       L->>L: Flan-T5 generation
20    end
21    L->>E: Complete synthetic data
22    E->>E: Statistical tests
23    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