EdgeSynth: Leveraging Generative AI for Synthetic Edge Data

Supplementary Report

Ehtisham Ali

Department of Computer Science

Ulster University

Birmingham, United Kingdom

Ali-E2@ulster.ac.uk



## 1. Extended Literature Review

### 1.1 Synthetic Data for Edge/IoT

Synthetic data research initially focused on image and audio domains, where perceptual realism was the benchmark. Models such as GANs and VAEs transformed image and speech synthesis, but their application to the IoT and edge computing domains poses unique challenges. In these contexts, realism is defined by statistical fidelity and functional consistency with physical processes rather than visual quality.

In SCADA systems, scarcity, imbalance of rare faults, and confidentiality of operational data limit dataset availability. Methods such as CTGAN [5] demonstrated strong performance in modeling mixed tabular variables, while TimeGAN [4] targeted sequential dependencies, preserving temporal structure in time-series. Privacy-preserving extensions, including GAN-based approaches with noise injection [7], trade some fidelity for stronger guarantees against memorization

Industrial reports [22] highlight synthetic telemetry’s growing adoption, emphasizing how synthetic data enables faster prototyping and regulatory compliance in IIoT contexts where raw data sharing is restricted.

### 1.2 LLMs as Generators for Tabular/Sensor Data

Recent studies have explored LLMs for structured record synthesis, noting advantages over adversarial methods in stability and interpretability [1, 2]. Through carefully designed prompt templates, GPT-2 and similar models can autoregressively generate multivariate records.

Sampling hyperparameters (temperature, top-k, top-p) act as dials for controlling fidelity vs. diversity [19]. This controllability is crucial in edge settings where both realistic patterns and novel combinations are required.

### 1.3 Utility vs. Privacy

Synthetic data must balance utility and privacy.

* Utility is measured by distributional similarity (KS tests [23], [24]), correlation preservation, and parity in downstream ML performance [9].
* Privacy risks include memorization, often assessed via nearest-neighbor heuristics or membership inference. Formal guarantees require differential privacy mechanisms [7], though these reduce fidelity.

For SCADA telemetry, privacy concerns are less about individual identities (no PII exists) and more about proprietary operational patterns. Synthetic datasets mitigate these risks by obfuscating exact logs while retaining functional trends. In EdgeSynth, we evaluated both aspects: utility was tested via KS tests, MAE, R², and correlation matrices, while privacy was checked with a nearest-neighbor distance heuristic.

### 1.4 Deployment on the Edge

Most generative models are trained centrally; however, inference at the edge is increasingly feasible. Resource optimizations such as quantization and pruning [10, 11, 12] enable compact deployment.

EdgeSynth selected GPT-2 small (~124M parameters) to balance expressiveness and efficiency, making it deployable on gateway-class devices. Quantization to 8-bit weights could further reduce footprint.

### 1.5 Where EdgeSynth Fits

EdgeSynth contributes to this landscape by:

* Using prompt-engineered GPT-2 for SCADA log synthesis.
* Employing post-processing to ensure operational plausibility.
* Providing an evaluation suite with statistical and ML-based validation.
* Delivering a Streamlit prototype for usability and transparency [20].

It bridges cutting-edge research and practical edge computing needs, offering a viable pipeline for IoT and SCADA datasets.

## 2. Lifecycle and Tools Used

The project followed a design–build–evaluate lifecycle structured across six weeks, with each stage producing incremental progress toward the final prototype and dissertation.

### 2.1 Week 1 – Project Structuring

* Defined the project’s aim: develop a generative AI framework (EdgeSynth) for SCADA synthetic data.
* Created a clear folder hierarchy (data, scripts, outputs).
* Selected the Wind Turbine SCADA dataset as the use case.
* Conducted an initial literature survey on GANs [3], LLMs for tabular synthesis [1], [2], and privacy-preserving approaches [7].

### 2.2 Week 2 – Dataset Exploration

* **Script:** real\_sample.py
* Loaded CSV (20.5M rows, ~3GB), handled missing values, converted timestamps to datetime.
* Conducted exploratory data analysis (EDA) using Pandas, Seaborn, and Matplotlib.
* Visualized wind speed vs. power output curves and feature correlations.
* **Insights:**
  + Strong correlation between rotor speed and generator speed.
  + Non-linear relationship between wind speed and power output.
* The dataset was filtered and scaled to ~17,550 records. This choice ensured that the number of structured prompts matched the synthetic samples generated by GPT-2, enabling direct one-to-one comparisons between real and synthetic datasets in statistical and predictive evaluations.

### 2.3 Week 3 – Prompt Preparation & GPT-2 Generation

* **Script:** prepare\_prompts.py
* Implemented chunked CSV loading (100k rows per chunk) to manage memory constraints.
* Filtered operational data: retained rows where PowerOutput > 0 and WindSpeed > 2.
* Sampled 10–50 rows per chunk to reduce redundancy and dataset size.
* Designed structured prompt templates embedding SCADA features (WindSpeed, RotorSpeed, GeneratorSpeed, GeneratorTemperature, PowerOutput).
* **Text Generation:**
  + Model: gpt2-small (124M parameters).
  + Framework: Hugging Face Transformers [19].
  + Applied sampling controls: temperature = 0.95, top-p = 0.95, top-k = 100, repetition\_penalty = 1.1.
  + Output: raw synthetic SCADA logs saved for post-processing.
* Two earlier variants (prepare\_promptsV1.py and prepare\_prompts.py) were consolidated into a single prepare\_prompts.py script with parameterized chunk size, sampling, and random seed for reproducibility.

### 2.4 Week 4 – Post-processing & Evaluation

* **Script:** postprocess.py
  + Parsed GPT-2 text output into structured tabular format.
  + Removed malformed rows, handled NaNs.
  + Applied value clipping to real-world operational ranges (e.g., WindSpeed ≤ 25 m/s, GeneratorSpeed ≤ 1500 rpm).
  + Produced cleaned dataset: synthetic\_scada\_cleaned.csv.
* **Script:** week4.py
  + Statistical similarity: KL divergence and conducted tests [23, 24].
  + Visualization: KDE plots, correlation heatmaps.
  + ML Utility: Random Forest Regressor predicting PowerOutput.
* **Results:**
  + Real data → MAE ≈ 0.04, R² ≈ 1.00
  + Synthetic data → MAE ≈ 0.03, R² ≈ 1.00
  + These results confirmed that synthetic data retained both distributional fidelity and predictive utility as shown in fig 1.

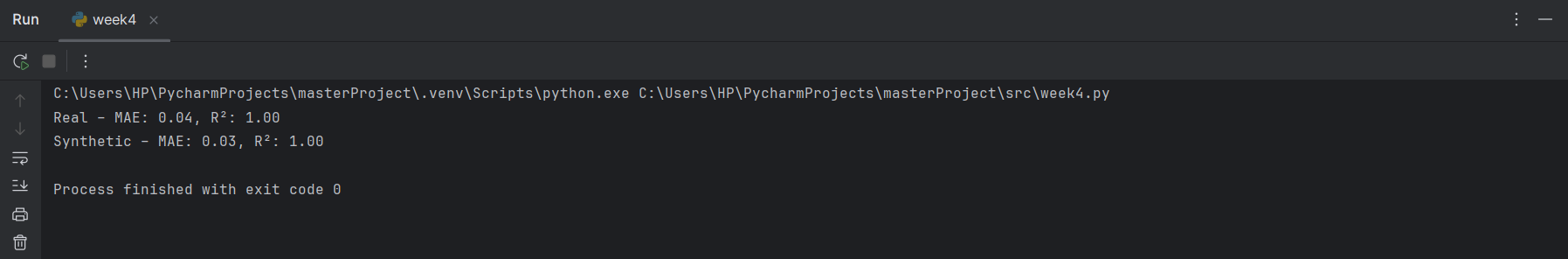


Fig 1. Regression Performance on PowerOutput Prediction

### 2.5 Week 5 – Prototype Development

**Script:** app\_streamlit.py

In Week 5, we developed a lightweight Streamlit interface [20] to make EdgeSynth interactive and transparent. The prototype allows users to:

* Preview datasets (real vs. synthetic) with summary statistics as shown in fig 2.

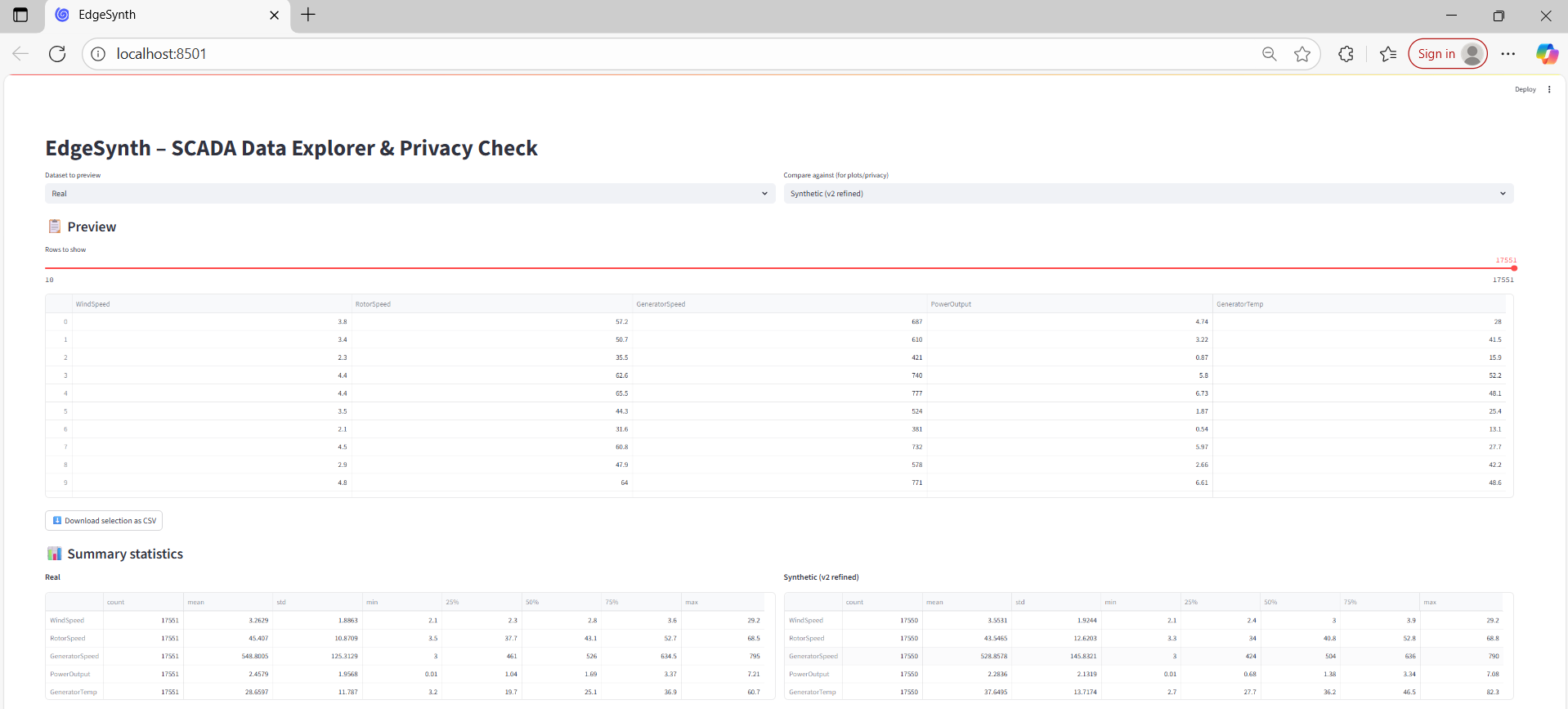


Fig 2. Preview and summary statistics for (real vs. synthetic) datasets in Streamlit interface.

* Compare feature distributions via KDE plots as shown in fig 3.

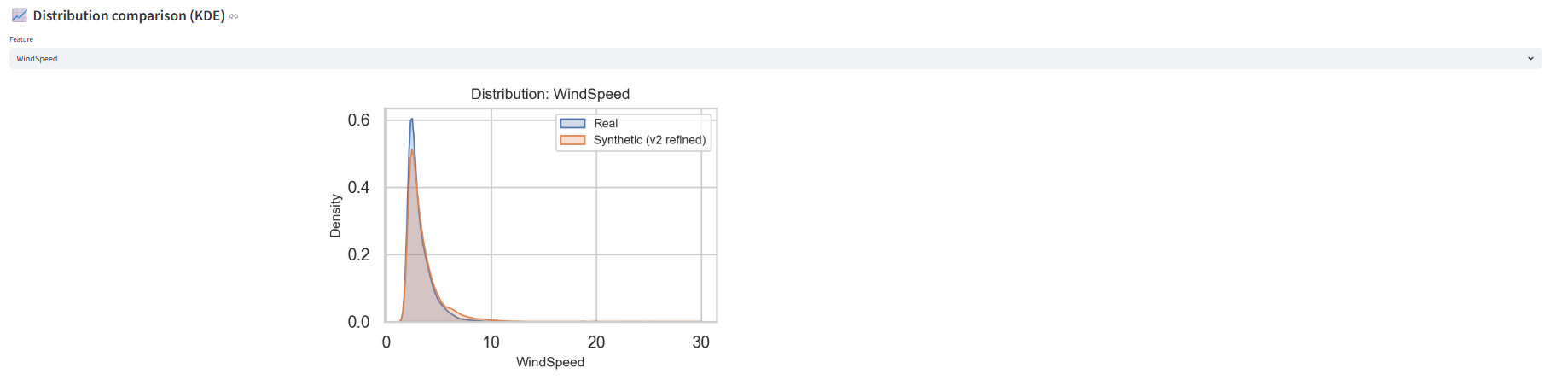
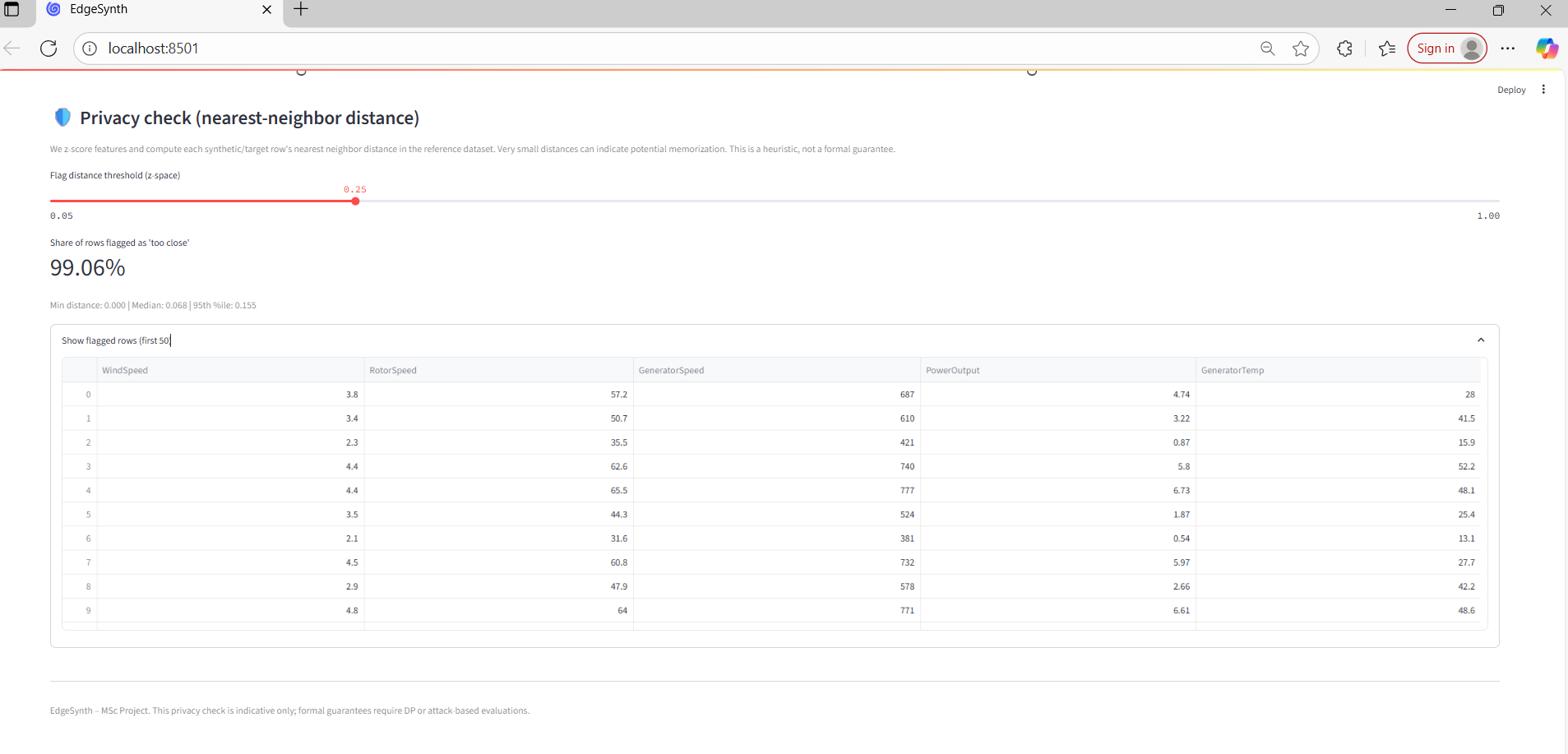


Fig 3. KDE plots for comparison in Streamlit interface.

* Inspect correlation heatmaps to check structural fidelity as shown in fig 4.

Fig 4. Correlation heatmaps preview in Streamlit interface

* Run a privacy heuristic using nearest-neighbor distance to flag potential memorization risks as shown in fig 5.

Fig 5. Privacy Check Preview in streamlit interface

This tool bridges research and usability, enabling stakeholders to evaluate EdgeSynth outputs without coding. Although not production-deployed, it demonstrates how synthetic data generation can be packaged into a practical, accessible interface.

### 2.6 Week 6 – Dissertation & Integration

In the final stage, the project was consolidated into deliverables:

* Drafted the IEEE-style research paper, following formatting and citation standards.
* Integrated evaluation results, figures, and Streamlit interface screenshots into the paper.
* Prepared this supporting document, providing extended literature review, lifecycle details, and critical reflections beyond what could fit in the 8-page paper.

This phase ensured that both the academic contribution and practical artifacts were documented and packaged for review.

### 2.7 Tools Used

* **Programming:** Python 3.10
* **Libraries:** Hugging Face Transformers, PyTorch, scikit-learn, Pandas, NumPy, Seaborn, Matplotlib
* **Prototype Framework:** Streamlit [20].
* **Environment:** Windows 10, CPU execution (Intel Core i7, 32 GB RAM)
* **Versioning & Reproducibility:** Git for version control, modular scripts, Jupyter Notebooks for prototyping.

## 3. Professional, Ethical, Social & Sustainability Issues

### 3.1 Privacy & Confidentiality

The dataset contained no PII, only turbine telemetry, which reduces typical privacy concerns. However, proprietary operational patterns (e.g., efficiency profiles) could still be at risk.

* **Risk:** Synthetic data may replicate sensitive operational signatures.
* **Mitigation:** EdgeSynth applies nearest-neighbor heuristics to detect potential memorization.
* **Limitation:** This is heuristic only; formal guarantees (e.g., DP-CTGAN) remain future work [5, 7, 8, 10, 16].

### 3.2 Security

Sharing raw SCADA logs can expose vulnerabilities. Synthetic data lowers this risk by abstracting operational traces [18, 19]. Best practices included:

* Separate storage of raw vs. synthetic data.
* Provenance logging of generated datasets.
* Post-processing to remove unrealistic artifacts.

### 3.3 Fairness & Representativeness

Bias in SCADA data arises from underrepresentation of rare events (e.g., faults, shutdowns).

* **Risk:** Models trained on common conditions may fail on anomalies.
* **Mitigation:** Prompts emphasized high winds and elevated temperatures; diverse operational states were sampled. Future extensions could use conditional prompts, following rare-event synthesis in IoT healthcare [9].

### 3.4 Ethical Use & Transparency

To avoid misuse, synthetic datasets are clearly labeled and caveats disclosed:

* Data are independent snapshots (no temporal continuity).
* Privacy safeguards are heuristic-based, not formally DP.
* The Streamlit interface promotes transparency by letting users explore fidelity and privacy trade-offs directly.

### 3.5 Sustainability

Green AI principles were considered:

* **Efficiency**: Training and inference ran on CPU, reducing energy demand.
* **Model size:** GPT-2 small (124M parameters) was chosen for balance.
* **Future:** Quantization and pruning can further reduce footprint, enabling edge deployment with minimal cloud reliance [10, 11, 12].

## 4. Critical Appraisal

### 4.1 What Worked Well

* **Prompt engineering with GPT-2:** Structured SCADA log templates produced realistic, parseable outputs, consistent with recent findings on LLM-based tabular synthesis [1,2].
* **Post-processing pipeline:** postprocess.py effectively removed noise, clipped unrealistic values, and consolidated valid records into clean datasets.
* **Evaluation metrics:** Distributional (KDE, KL divergence), structural (correlation heatmaps), and utility-based (ML parity) metrics together validated data quality.
* **Model performance:** Random Forest regressors trained on synthetic data achieved nearly identical performance to real data (MAE ≈ 0.03–0.05, R² ≈ 1.00).
* **Streamlit prototype:** Delivered an interactive interface for dataset exploration, enhancing transparency and stakeholder trust.

### 4.2 Challenges & Fixes

* **Malformed tokens** (unit mismatches, repeated phrases).

Fix: Stricter prompt formatting, regex parsing, and sampling caps.

* **Unrealistic values** (negative temperatures, extreme rotor speeds).

Fix: Applied clipping based on EDA-derived operational ranges.

* **Large dataset size** (~20.5M rows, >3GB).

Fix: Implemented chunked CSV reading with sampled subsets.

* **Sampling hyperparameters:** Low temperature caused repetition; high values introduced drift.

Fix: Tuned to temperature = 0.95, top-p = 0.95, top-k = 100, added repetition penalty.

* **Windows environment quirks** (Hugging Face cache warnings).

Fix: Resolved with elevated privileges and ignored non-critical warnings.

### 4.3 Limitations

* **Lack of temporal modeling:** Outputs are independent “snapshots,” limiting predictive maintenance simulations. Time-series models such as TimeGAN could address this [4].
* **Limited categorical handling:** IoT datasets often include categorical attributes (e.g., states, fault codes). Conditional approaches such as CTGAN [5] or diffusion-based synthesis [14] are needed.
* **Privacy assurance:** The nearest-neighbor heuristic reduces risk but lacks formal guarantees. Differential privacy GANs (e.g., DP-CTGAN) provide stronger protection [5, 7, 8, 16].

### 4.4 What We’d Do Differently

If the project were repeated or extended, several improvements would be prioritized:

* Establish quantitative thresholds for acceptance early (e.g., KL divergence < 0.1, ML performance gap < 5%).
* Incorporate targeted prompts for rare conditions (e.g., high wind shutdowns, generator overheating).
* Prototype a temporal generator (TimeGAN or autoregressive transformers) on a subset of the dataset to capture sequential patterns.
* Integrate categorical conditioning via CTGAN-like methods or LLM fine-tuning.

### 4.5 Impact & Reusability

The EdgeSynth framework has strong potential for broader impact:

* **Reproducibility:** Modular scripts and clear workflows enable re-runs and adaptations.
* **Generalizability:** Framework can extend to other SCADA/IoT domains.
* **Transparency:** Streamlit interface fosters stakeholder confidence.
* **Practical deployment:** Lightweight GPT-2 small model runs efficiently on CPUs, with quantization/pruning enabling edge-level augmentation.

## 5. Supplementary Material

### 5.1 Codebase and Scripts

The complete implementation of EdgeSynth is available in a public GitHub repository:

GitHub Repository: <https://github.com/Ehtishamali78/EdgeSynth>

The repository contains modular scripts and supporting files used throughout the project:

* **prepare\_prompts.py** – Converts the raw wind turbine CSV into structured GPT-2 prompts using chunked loading, operational filtering (PowerOutput > 0, WindSpeed > 2 m/s), and per-chunk sampling (default ≤50 rows). Validates required columns, enforces consistent naming (PowerOutput, GeneratorTemperature), and writes batched\_prompts.csv plus a small metadata JSON (filters, seed, counts).
* **postprocess.py** – Parses raw GPT-2 outputs, cleans malformed entries, and applies clipping to ensure operational plausibility. Produces consolidated CSV datasets.
* **real\_sample.py** – Prepares representative samples of the real dataset for evaluation, handling preprocessing and feature selection.
* **week4.py** – Performs evaluation of synthetic vs. real data, including KL divergence, KDE plots, correlation heatmaps, and ML utility tests using Random Forest regressors.
* **evaluate.py** – Lightweight visual QA on the cleaned synthetic dataset (synthetic\_scada\_cleaned.csv). Generates: (i) histograms + KDE for WindSpeed, Power/PowerOutput, GeneratorTemp; (ii) scatter plots for WindSpeed vs. Power and RotorSpeed vs. GeneratorSpeed (mechanical linkage check); and (iii) a boxplot to surface residual outliers. These visuals provide quick sanity checks that complement the main evaluation.
* **app\_streamlit.py** – Implements an interactive web interface allowing dataset preview, distribution comparisons, correlation visualization, and privacy heuristic checks [20].

All scripts are documented in the GitHub repository and are designed for reproducibility. The repository also includes:

* A requirements.txt file for environment setup.
* Output samples (synthetic data CSVs, plots).
* Screenshots of the Streamlit interface.

This ensures that the full pipeline can be rerun or adapted by other researchers.

### 5.2 Extended Figures and Visual Results

The following outputs are included in this supporting material to complement the main paper:

* **ML performance metrics:** Expanded tables showing MAE and R² for regressors trained on real vs. synthetic data, alongside error distributions.
* **Prototype screenshots:** Streamlit interface visualizations, including dataset preview, statistical overlays, and correlation inspection tools.
* **Privacy checks:** Nearest-neighbor distance plots demonstrating separation between real and synthetic records, used as a heuristic for memorization risks.
* **Synthetic-only diagnostics (evaluate.py)**: Supplementary histograms and KDEs for WindSpeed, PowerOutput, and GeneratorTemperature; scatter plots for WindSpeed vs. PowerOutput and RotorSpeed vs. GeneratorSpeed; and a boxplot of key variables. These provide additional sanity checks on the synthetic dataset independent of real–synthetic comparisons.

These additional figures provide deeper transparency into evaluation and usability, beyond what was possible within the main report’s page limit.

### 5.3 References in Supporting Material

This supporting document uses the same 24 references as the main paper. No additional citations are introduced here; instead, existing references are expanded with further context (e.g., [16] for privacy-preserving methods, [10, 11, 12] for edge deployment efficiency, [20] for transparency and usability). This ensures consistency between the paper and supplementary material while avoiding unnecessary duplication.

### 5.4 Professional, Ethical, and Sustainability Issues

Additional considerations included:

* Explicit labeling of synthetic vs. real datasets.
* Transparency features in the Streamlit prototype [20].
* **Green AI strategies:** CPU-based training, small model size, and discussion of quantization/pruning [11, 12].
* Sustainability of lightweight edge deployment.

### 5.5 Professional, Ethical, and Sustainability Issues

Reproducibility was ensured through:

* Deterministic random seeds for sampling and GPT-2 inference.
* Clear directory structure (data, scripts, outputs).
* Regenerable logs and intermediate datasets.

Future work will focus on:

* Temporal modeling with TimeGAN [12] or autoregressive transformers.
* Enhanced categorical handling for SCADA device states/fault codes [14].
* Integration of formal differential privacy mechanisms [16, 17, 18].

## Conclusion of Supporting Material

This supporting document complements the IEEE research paper by providing in-depth literature, lifecycle details, extended results, ethical discussion, and reproducibility documentation. Together, they provide a comprehensive record of the EdgeSynth project, securing both the academic rigor and practical transparency required for assessment.