International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 11 November-2026 | Impact Factor: 3.5

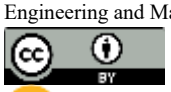


AI-Based Energy Theft Detection and Smart Grid Analytics Platform

Aryan Gajbhare[1], Himanshu Rodge[2], Sarthak Dhoke[3], Aryan Battu[4], Shivraj Shinde[5]
[1–5] Department of Computer Science and Engineering (AI & ML), Jhulelal Institute of Technology, Nagpur, Maharashtra, India
Corresponding Author Email: aryangajbhare716@gmail.com

How to Cite this Article:
Gajbhare, A., Rodge, H., Dhoke, S., Battu, A. & Shinde, S. (2026). AI-Based Energy Theft Detection and Smart Grid Analytics Platform. International Journal of Creative and Open Research in Engineering and Management, 02(11), 1-9. https://doi.org/10.55041/ijcope.v2i11.117

License:
This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.
© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.







https://doi.org/10.55041/ijcope.v2i11.117

Abstract—

Electricity distribution utilities must examine large volumes of smart-meter readings to find consumption patterns that may indicate non-technical losses, yet unusual readings can equally arise from changes in occupancy, seasonality, solar generation, meter faults or data errors. Published work concentrates mainly on detection models, whereas the workflow around data quality, explanation and human review is less often documented as one reproducible system. This paper presents GridIntel, a web-based platform that integrates comma-separated-value (CSV) data ingestion, data-quality auditing, sixteen customer-level statistical features, feeder-month energy-balance analysis, Isolation Forest anomaly scoring and, when sufficient labels exist, Random Forest, XGBoost and ensemble scoring. Scores are expressed as cohort-relative percentile rankings with feature-based explanations and feed an investigation workflow with case snapshots, an append-only audit trail and CSV and portable-document-format (PDF) reports. The platform also provides a separate past-bill analysis using browser-side optical character recognition (OCR) and a dataset-grounded assistant. It is implemented with React, TypeScript, FastAPI and Python analytics libraries. In a recorded verification run, 263 of 264 backend tests and all 17 frontend unit tests passed, one opt-in live-Ollama test was skipped, the production build succeeded, and a manual 12-route browser smoke test showed no console errors or warnings. No independent labelled utility dataset was used, so no claim is made about real-world detection accuracy. The contribution is an integrated software architecture and investigation workflow, not a new learning algorithm.

Keywords— Smart Grid Analytics; Non-Technical Loss Detection; Electricity Theft Investigation; Anomaly Detection; Explainable Machine Learning; Data Quality Assessment.

I. INTRODUCTION

Part of the energy that a distribution utility puts into its network is never billed. Technical losses are the portion dissipated as heat in conductors, transformers and other network equipment, and they can be explained from the characteristics of the network itself. Non-technical losses (NTL) make up the rest. They include electricity theft and meter tampering, but also faulty or inaccurate meters, unmetered supply, and mistakes in meter reading and billing [1], [2]. Theft is therefore only one possible source of NTL, and metering faults or billing errors should not be mistaken for it.

Smart meters produce fine-grained consumption records, but analysing them at scale is hard. Real datasets contain missing, duplicated and inconsistent readings, and an unusual value may result from a communication failure or a data-handling error rather than from anything a customer did. Reviews of smart-meter analytics separate such bad data from energy theft, since each calls for a different detection approach [3]. A change in consumption may also reflect a different household occupancy, the season or rooftop solar generation, so an anomaly in the readings cannot by itself reveal its cause.

Machine-learning methods for NTL detection usually learn consumption patterns from historical readings and frequently use inspection results as labels [4]–[8]. Workflows built on these models have several practical limits. Labels from inspections are expensive to collect, scores add little if the underlying data cannot be trusted, and an inspector can do little with a score that comes without an explanation. A report on an operational NTL system found that explaining detected patterns and having people validate them is practically indispensable [2].

For these reasons, a high anomaly score should trigger a human review, not an accusation. The



score places a customer relative to the others in the same dataset. It is not the probability that theft took place, and the same pattern may have an innocent explanation. Any conclusion about a particular customer has to be checked against bills, meters and conditions in the field.

Much of the literature centres on the detection model and reports how it performs on a utility dataset. Fewer papers describe, in a reproducible way, the workflow an analyst needs around the model: checking the data before analysis, ranking customers with explanations, testing the energy balance at feeder level and recording what happens to each lead. GridIntel was built to combine these steps in a single web-based platform.

This paper presents the design and implementation of GridIntel and the software verification performed on it. The contribution is an integrated software architecture and investigation workflow; the paper neither proposes a new learning algorithm nor reports independently validated detection accuracy. The objectives are:

- to read consumption data from CSV files and audit its quality before analysis;
- to compute sixteen explainable customer-level features without filling unavailable values with zero;
- to rank customers with unsupervised scoring and, when enough labels exist, supervised scoring, expressed as cohort-relative percentile bands with feature-based explanations;
- to compare feeder input energy with billed energy month by month;
- to manage investigation cases with preserved score snapshots, an append-only audit trail and CSV/PDF reports, and to provide a separate past-bill analysis and a dataset-grounded assistant; and
- to document the verified software behaviour together with its limits.

The remainder of the paper is organised as follows. Section II reviews related work and identifies the research gap. Section III describes the methodology and implementation of GridIntel. Section IV presents the software verification, discussion and limitations. Section V concludes and outlines future work.

II. LITERATURE REVIEW

Work on NTL detection spans surveys, supervised and unsupervised detectors, deep-learning models and, more recently, studies of explanation and of deployment in operating utilities. The studies discussed below are representative and are summarised in Table I.

Glauner et al. [1] surveyed how artificial intelligence has been applied to NTL detection. They group the approaches into expert systems and machine learning and go through the algorithms, features and datasets in use as well as the challenges that remain open.

Among early supervised studies, Nagi et al. [4] trained a support vector machine on irregularities in historical kWh records from three towns in peninsular Malaysia to choose which customers to inspect on site; they reported that the utility's inspection hit rate would increase from 3% to 60%. A later study by the same group [5] placed a fuzzy inference system after the SVM stage and reported a further increase from 60% to 72%.

Jokar et al. [6] used two levels of analysis: meters on distribution transformers flag areas where theft is likely, and abnormal consumption patterns are then used to identify suspicious customers within them. They tested the approach on a real dataset of 5000 customers. Comparing supply-side with customer-side energy is also the idea behind the feeder analysis in GridIntel, although GridIntel does not separate technical from non-technical loss.

Zheng et al. [7] designed a wide and deep convolutional neural network. Its deep branch works on a two-dimensional representation of the data, in which theft appears as a break in the regular periodicity of normal use, whereas the wide branch captures global characteristics of the raw one-dimensional series. Buzau et al. [8] built a supervised method for industrial and commercial customers of the Spanish utility Endesa, drawing on smart-meter consumption, alarm records, electrical magnitudes and auxiliary databases. About 57,000 field inspections were used to train, validate and test the model, which



was then applied in a campaign to detect NTL among large customers.

More recent studies stay with the same families of models but pay more attention to class imbalance, interpretation and real utility data. Monteiro et al. [9] paired an Isolation Forest anomaly score with an XGBoost classifier on consumption data from a Peruvian utility, handled class imbalance with SMOTETomek resampling, and reported an AUC-ROC of 0.999 and an F1-score of 0.77 at a decision threshold of 0.6. Their SHAP analysis highlighted extreme monthly variation, prolonged low consumption and tariff category as the most influential predictors. Badr et al. [10] reviewed data-driven approaches to electricity fraud detection in smart-metering systems. The figures reported in [8] and [9] come from the authors' own datasets and cannot be compared with the verification results in this paper.

Explanation and human review have also been studied in operating utilities. Coma-Puig et al. [2] report a case study in which explainable AI, including Shapley-value analysis, was added to a mature NTL detection system already running at a utility; they observe that detected patterns almost always need validation by people, which is difficult without explanations. Wang et al. [3] reviewed smart-meter data analytics and drew a line between bad data and energy theft. Missing readings and other data anomalies can arise in communication networks, metering devices and servers [17], and they have to be separated from genuine anomalies before scores are interpreted.

The algorithms used in GridIntel are well established. Isolation Forest [11] works without labels: it splits the feature space at random, and unusual observations tend to be separated after fewer splits than normal ones. Random Forest [12] combines the predictions of many decision trees, and XGBoost [13] is a scalable implementation of boosted trees; both are supervised and therefore depend on labelled data. SHAP [14] explains an individual prediction by assigning each input feature an additive contribution. When positive cases are rare, precision–recall measures usually say more than ROC curves alone [15]. The implementation uses scikit-learn [16].

Table I: Comparison of Representative Studies with GridIntel

Author and year	Research approach	Main contribution	Relevant limitation	Relevance to GridIntel
Glauner et al. (2017) [1]	Survey of expert-system and machine-learning methods	Reviews methods, features, datasets and open problems in NTL detection	Survey only; no implemented investigation workflow	Supplies NTL terminology and the challenges behind label-aware scoring
Nagi et al. (2010) [4]; (2011) [5]	SVM, then SVM with fuzzy inference	Selects suspect customers for inspection from historical kWh data; reports higher hit rates at a Malaysian utility	Tested on one utility's data; supervised and label-dependent	Supports consumption-pattern features and score-based prioritisation of inspections
Jokar et al. (2016) [6]	Consumption-pattern detector with transformer-meter area shortlisting	Two-level analysis; tested on a real dataset of 5000 customers	Requires transformer-level metering	Related to the feeder-level energy-balance analysis
Zheng et al. (2018) [7]	Wide and deep convolutional neural network	Combines two-dimensional periodicity features with global one-dimensional features	Deep model; needs labelled data and offers limited transparency	Contrasts with transparent statistical features and rank-based scoring
Buzau et al. (2019) [8]	Supervised learning on smart-meter, alarm and auxiliary data	Trained and tested on about 57,000 field inspections; used in a campaign for large customers	Industrial and commercial customers; depends on inspection labels	Shows the value of real inspection outcomes, which GridIntel case outcomes could supply in future
Wang et al. (2019) [3]	Review of smart-meter data analytics	Separates bad data from energy theft in anomaly detection	A review; no system	Motivates auditing data quality before scoring
Coma-Puig et al. (2024) [2]	Case study of explainable AI in an operational NTL system	Shapley-value explanations; stresses human validation	Describes one company's system	Closest in spirit; GridIntel adds a quality audit, feeder balance and an audit trail
Monteiro et al. (2025) [9]	Isolation Forest, XGBoost and SHAP on real utility data	Hybrid detection with reported AUC-ROC 0.999 and F1 0.77	Model-centred; results specific to its dataset and labels	Same model families; GridIntel combines rank-normalised scores instead of stacking



Traditional detection studies treat the problem as a modelling task: a dataset is provided, a classifier or anomaly detector is trained, and results are reported with metrics such as AUC or F1. A system meant to support investigations has to answer further operational questions: whether the input data can be trusted, why a customer ranked highly, what was done about each lead, and how the outcome was recorded. The studies in [2] and [3] show that explanation, human validation and the separation of bad data from theft all matter in practice. GridIntel belongs to this second group.

Across the studies reviewed here, detection models are mature. What is offered less often, in a form that can be reproduced locally, is a documented software workflow that brings data-quality auditing, label-aware scoring, feature-based explanation, feeder energy-balance analysis and audited case management together in one system. GridIntel addresses this gap at the level of software design. The review is not exhaustive, and the paper does not claim that GridIntel is the first system of its kind.

III. METHODOLOGY

This section describes what GridIntel implements. The pipeline rules, statistical definitions and scoring thresholds that the codebase applies are documented below.

A. Overall System Architecture

GridIntel follows a client–server architecture (Figure 1). The front end is a React and TypeScript application built with Vite and styled with Tailwind CSS, with routing handled by React Router and charts drawn with ECharts. It talks to a Python back end based on FastAPI, using Pydantic to validate requests, SQLAlchemy for database access, Alembic for schema migrations and Uvicorn as the server. The analytics layer relies on pandas, NumPy, scikit-learn [16], XGBoost [13], SHAP [14] and joblib. Datasets, run results and cases are kept in a local database, trained models are saved as artifacts, and ReportLab generates the PDF and CSV reports. Photos of past bills are read by an OCR engine that runs in the browser as WebAssembly, and an optional local Ollama model supports the assistant. The interface checks a back-end health endpoint every 15 seconds. The technologies are listed in Table II.

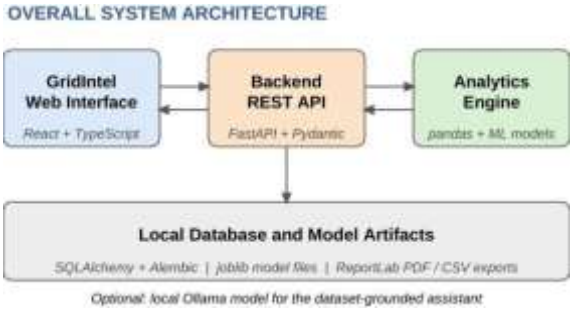


Figure 1: Overall GridIntel system architecture.

Table II: Technology Stack

Layer	Technology
Front end	React, TypeScript, Vite, Tailwind CSS, React Router, ECharts
Back end	Python, FastAPI, Pydantic, SQLAlchemy, Alembic, Uvicorn
Data and ML	pandas, NumPy, scikit-learn, XGBoost, SHAP, joblib
Reports	ReportLab
Bill OCR	Tesseract-based English OCR in the browser (WebAssembly)
Optional assistant	Local Ollama language model

B. System Workflow

A typical session follows Figure 2. The user uploads a CSV file or generates a synthetic dataset, after which the data-quality audit is computed. Customer features are then calculated and, where feeder input data exist, so is the feeder-month balance. A model run trains the supported models and ranks the customers; the user then reviews the leads together with their reasons, opens investigation cases and exports reports. Each dataset-based page works on the active dataset chosen in the top bar, and when data are missing the page shows an empty state or a dash instead of making up numbers.

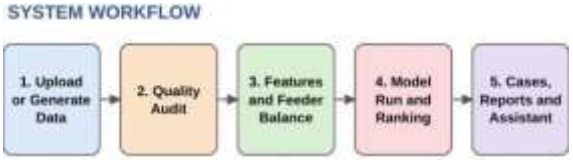


Figure 2: GridIntel processing workflow.

C. Data Ingestion

Two CSV layouts are accepted. In the long layout every row is a single reading with a customer identifier, a timestamp or date and a consumption value in kWh. In the wide layout every row is one customer and each daily date forms a column. Feeder, transformer, customer-type and label fields are optional in both layouts. Before anything is imported, the back end returns a preview showing the detected layout, the



columns, a suggested column mapping, warnings and the first rows; the user can correct the mapping and then names the dataset. Import runs in chunks, and the test suite covers chunked ingestion and rollback. Timestamps in standard ISO (YYYY-MM-DD) and regional (DD/MM/YYYY) formats are parsed into normalised daily calendar dates, while non-numeric consumption entries and unparseable timestamps are marked as invalid records at the preview stage.

D. Synthetic Dataset Generation

To test the pipeline, the platform can generate reproducible synthetic datasets with 10 to 20,000 customers, 14 to 730 days and 1 to 250 feeders. The user chooses a suspicious-pattern ratio between 0 and 50% and a fixed random seed. Each customer's readings combine a Gaussian baseline with seasonality and meter noise. Customers chosen to behave anomalously receive an injected pattern: a sudden drop, a gradual decline, intermittent zeros, night-time-only usage or a flatlined reading, which the interface describes as seeded consumption cliffs. The synthetic labels record which customers received an injected pattern; they are ground truth about the simulation and not verified theft cases. Such data are useful for checking that the pipeline runs end to end and behaves deterministically, but they cannot show that the models would detect real theft, because the injected patterns follow the designer's assumptions. Each baseline series is generated as

$$x_t = \max(0, \mu_i + A_i \sin(2\pi t/365) + \varepsilon_t)$$

with Gaussian noise $\varepsilon_t \sim \mathrm{N}(0, \sigma_i^2)$. Anomalous customers receive multiplicative step-drops of 40–80%, linear decay slopes or zero-injection masks that start on a randomised onset day.

E. Data Quality Assessment

After import, the audit applies the six checks listed in Table III and reports a quality score from 0 to 100. It also reports interval coverage (readings received against readings expected), the number of customers whose coverage is below 90%, and a filterable list of issues with severity (critical, warning or information), the entity affected and a sample value. Missing readings are never treated as zero consumption.

Table III: Data Quality Checks and Weights

Check	Issue examined	Weight
Missing values	Null customer ID, timestamp or kWh	35%
Duplicate readings	Same customer and timestamp	20%
Negative values	Impossible negative readings	15%
Extreme outliers	Outside IQR or robust bounds	15%
Long zero runs	Three or more consecutive zero days	10%
Time-series gaps	Expected dates without a reading	5%

The score is defined as

$$Q = 100 \times \left(1 - \sum_{k=1}^{6} w_k p_k\right), \quad \sum_{k=1}^{6} w_k = 1$$

where w_k are the weights in Table III and $p_k \in [0, 1]$ is the penalty of check k. Each penalty is the ratio of flagged records (or affected customer-days) to the total expected records for that check; a check whose violation share exceeds 5% is marked critical, 1–5% warning, and below 1% informational. The weights are fixed design parameters and have not been calibrated empirically in this work. The score describes the quality of the input data; it is neither the probability of theft nor a measure of model accuracy, and low completeness weakens every downstream feature.

F. Customer Feature Engineering

Sixteen explainable features are computed for every customer from the daily readings (Table IV). A feature that cannot be computed from the available readings is stored as missing and displayed as a dash; it is never replaced by zero. A ratio is undefined when its denominator is zero, and a windowed feature needs enough observations to fill its window. The data_completeness feature records how much of the expected calendar was observed, so reviewers can judge how far the other features can be trusted for an incomplete series. Window lengths, minimum observation counts, the peer-group definition and the handling of zero denominators are set in the back-end feature registry. Rolling and comparative features require at least 14 valid daily observations (7-day block comparisons need two full consecutive blocks), peer deviation is normalised against the median of the same customer category, and any division by zero returns a missing (NaN) value.



Table IV: Customer-Level Features (Documented Meaning)

No.	Feature	Definition as documented	Interpretation
1	mean_kwh	Mean daily consumption, $\mu = (1/n)\,\Sigma\, x_t$ over the recorded days	Typical level of daily usage
2	std_kwh	Standard deviation σ of daily consumption	Spread of daily usage
3	cv	Coefficient of variation, σ/μ (undefined when $\mu = 0$)	Spread relative to the customer's own average
4	min_kwh	Lowest daily consumption	Lower bound of observed usage
5	max_kwh	Highest daily consumption	Upper bound of observed usage
6	median_kwh	Middle value of daily consumption	Robust typical level
7	zero_ratio	Recorded days with zero use divided by recorded days	Share of zero-use days
8	longest_zero_run	Longest streak of consecutive calendar days with zero use	Prolonged zero consumption
9	flatline_score	Share of consecutive non-zero days with identical totals	Repeated identical readings
10	recent_vs_history_ratio	Recent 30-day usage relative to earlier history	Recent change in level
11	max_week_over_week_drop_pct	Largest fall between consecutive 7-day blocks, in per cent	Sudden reduction in weekly usage
12	trend_slope	Direction and rate of the long-run usage trend	Gradual rise or decline
13	weekday_weekend_ratio	Average weekday use relative to weekend use	Weekly usage pattern
14	self_deviation_max	Largest 7-day deviation from the customer's own baseline	Departure from own normal behaviour
15	peer_deviation	Average usage relative to same-type peers	Departure from comparable customers
16	data_completeness	Days with a reading divided by expected calendar days	Reliability of the other features

G. Customer-Level Consumption Visualisation

The customer page plots the daily kWh readings together with a rolling mean over seven readings. It also displays a note when the mean of the last 30 readings is at least 30% below the mean of the earlier readings, where the earlier readings are all valid daily observations before the most recent 30-day window. This note is a deterministic display rule, not a machine-learning prediction, and it does not by itself indicate theft.

H. Machine-Learning Models

Four scoring approaches are supported (Figure 3). Isolation Forest [11] is unsupervised: it partitions the feature space at random and scores a customer by how easily that customer is isolated, so no labels are required. Random Forest [12] and XGBoost [13] are supervised and need labelled data. The ensemble combines rank-normalised Isolation Forest scores with the supervised probabilities. Supervised training is attempted only when the dataset contains at least 30 positive labels. This eligibility threshold was chosen in the implementation; it is not a statistical guarantee that 30 positives suffice for reliable learning. When the condition is not met, the supervised models are skipped with a displayed reason, no labels are invented and the unsupervised score remains available. An unlabelled dataset therefore receives only unsupervised scores.

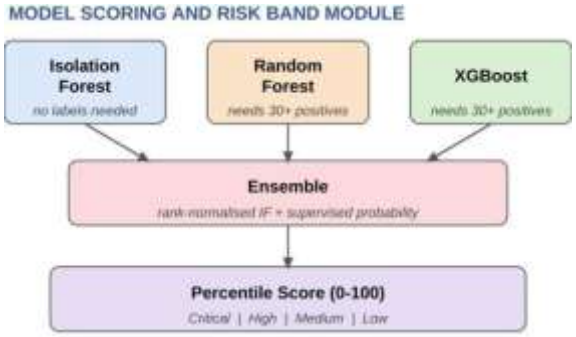


Figure 3: Model scoring and risk band module.

I. Training and Evaluation Methodology

Customers are divided into training and test partitions at customer level, by default 70% and 30%, using a configurable random seed. Pre-processing is fitted on the training partition only and then applied to the held-out partition. Since the split is by customer, no customer appears in both partitions. Leakage can still arise indirectly, however: a feature such as peer_deviation depends on other customers' data, and because peer-group statistics are computed over the full dataset during feature engineering, before the train–test split, information from test customers can influence training features. The documented protocol is a random customer-level split. A temporal split, which trains on an earlier period and tests on a later one, would be a stricter test of generalisation over time and is not part of the documented protocol.



For labelled runs, the implementation reports ROC-AUC, PR-AUC, precision, recall, F1-score, false-positive rate, the confusion matrix, precision and recall within the top 5% and 10% of ranked customers, the score distribution, the ten most important features, and comparisons with random ranking and with a naive recent-usage-drop rule. With true positives TP, false positives FP, false negatives FN and true negatives TN,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \quad \text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Positive cases are rare in NTL data, so a classifier can achieve a high ROC-AUC while still raising many false alarms; PR-AUC is reported because it reflects precision at low prevalence [15]. A trained model can also be evaluated on a second dataset. Runs on unlabelled data show score distributions only. Metrics obtained on synthetic datasets verify the pipeline under controlled conditions and are not evidence of real-world performance. No external labelled utility dataset was available for this paper, and no metric values are reported.

J. Risk Ranking

Each anomaly score is converted to a percentile rank r between 0 and 100 within the customer cohort of the dataset, and the rank defines the bands of Table V. The percentile is a relative position in the cohort and not a calibrated probability of theft; a customer in the Critical band is merely among the most unusual 2% of that cohort. In a cohort of 100 customers the Critical band holds about two customers, so very small cohorts give coarse bands, and the share of customers in the top band is fixed by construction whether or not any customer is genuinely anomalous. Tied anomaly scores receive average percentile ranks (method='average'). In the interface the output is called an anomaly priority or investigation lead.

Table V: Percentile-Based Risk Bands

Risk band	Percentile range	Cohort share
Critical	At or above the 98th	Top 2%
High	93rd up to below the 98th	Next 5%
Medium	80th up to below the 93rd	Next 13%
Low	Below the 80th	Remaining 80%

K. Explainability

For each score the system stores plain-English reasons tied to feature contributions. For the tree-based models, the contributions come from SHAP [14], which attributes a prediction to features additively. The customer page lists the leading anomaly signals with rank, direction, contribution, feature name and observed value. A deterministic rule such as the 30-day note differs from a model-based contribution: the rule is fixed in advance, whereas a contribution depends on the trained model and the data. SHAP values describe how the model uses its inputs; they are not causal explanations of customer behaviour. When tree-based SHAP values are unavailable (for example in pure Isolation Forest runs), fallback explanations use standardised feature z-scores relative to the cohort distribution.

L. Feeder-Level Energy Analysis

Where feeder input-energy records exist, GridIntel aggregates energy by calendar month and compares the feeder input with the sum of customer billed or read energy:

$$L_{f,m} = E^{\text{in}}_{f,m} - E^{\text{billed}}_{f,m}, \quad z_{f,m} = \frac{L_{f,m} - m_L}{s_L}$$

where f is a feeder, m a month, and m_L and s_L are robust location and scale estimates of the loss. A feeder-month is flagged when z exceeds 2.0, where

$$m_L = \text{median}(L_{f,\cdot}), \quad s_L = 1.4826 \times \text{MAD}(L_{f,\cdot})$$

(z is set to 0 when $s_L = 0$), and the loss percentage is computed as

$$\frac{L_{f,m}}{E^{\text{in}}_{f,m}} \times 100$$

Customers are mapped to feeders through the optional feeder field, so customers without a feeder identifier are not counted. Missing customer readings reduce the billed total and can make the loss look larger. The difference can be positive or negative and combines technical losses, unmetered consumption, metering errors and possible theft. It is a total difference, not confirmed electricity theft, and the system does not separate its components.



M. Investigation Management

Cases can be created from the leads table, from a customer page, or in bulk for the High and Critical leads of the latest completed run (up to 500 cases, skipping customers who already have an open case). Every case keeps a snapshot of the score and reasons from the time it was opened. The snapshot preserves the evidence on which the decision rested, so later model runs do not change it.

A case moves through the statuses shown in Figure 4. A case in review can return to open, and a case with a scheduled visit can return to review. Closing requires one of three outcomes: no issue found, issue found or inconclusive. A closed case can be reopened, which clears the outcome and the close time. Reviewers can set an assignee and a severity and add notes of 1 to 5000 characters. Every action is written to an append-only audit trail that covers case creation, changes of status, assignee and severity, notes, outcomes, reopening, report generation and bulk creation. "Append-only" here describes the behaviour of the application, which provides no way to edit or delete an event; it is not a claim of tamper-proof storage. The application has no login and the acting name is a self-entered setting, so the trail records attribution and not verified identity.



Figure 4: Investigation case lifecycle.

The Investigations page summarises lead quality from closed cases only. For each risk level it gives the share of issue-found and no-issue-found outcomes with sample counts and a warning that small samples are unreliable. Outputs are a queue CSV that follows the current filters (up to 50,000 cases), a CSV of run scores, a dataset summary PDF and a case PDF that contains the customer, risk snapshot, status, assignee, reasons, notes and the full audit trail.

N. Past Electricity Bill Investigation

A separate page compares the past bills of a single account and does not depend on the active dataset (Figure 5). Bills are supplied as photos or as a CSV file within the limits of Table VI. English OCR runs in the browser as WebAssembly. According to the project documentation, bill images are not sent to an OCR service, stored in the database or included in the analysis request, and only the values confirmed by the user are sent to the server.

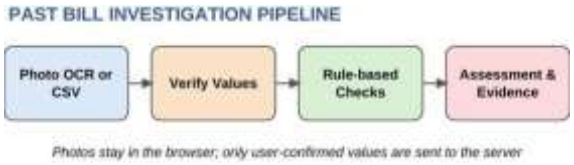


Figure 5: Past bill investigation pipeline.

Extracted or imported values appear in an editable table of billing periods with start, end, billed units, amount, solar export, source and confidence (high, medium, low or manual). Rows are checked for date order, units, duration and overlap, and a currency and a tariff rate per kWh can be added. Since OCR can misread digits and dates, the analysis runs only after the user confirms that the account identifier, dates, units and amounts match the original bills; OCR accuracy is not claimed.

The analysis applies deterministic checks and returns one of four assessments: anomalous, unusual pattern, no clear anomaly or insufficient evidence. The result includes a confidence level, evidence signals marked suspicious, watch or info, and a period table with days, billed, export and net kWh, average kWh per day, change, amount and effective rate. Net units are billed units minus solar export, the effective rate is the amount divided by units, and drops of 25% or more are highlighted. Seasonal variation, occupancy changes, solar generation, meter faults and billing corrections can all explain a change, and the tool cannot tell them apart from irregular use. The result is an investigation lead and not proof of theft.

Table VI: Past Bill Inputs and Limits

Item	Specification
Photos	2–12 images (JPG, PNG, WebP); 12 MB each, 30 MB total
CSV file	Up to 12 MB
Required columns	period_start, period_end, billed_units_kwh
Optional columns	account_id, bill_amount, solar_export_kwh
Date formats	YYYY-MM-DD or DD/MM/YYYY
Billing periods	2 to 36, no overlaps



O. Dataset-Grounded Assistant

An optional assistant answers questions about the stored records of the selected dataset through deterministic database tools. A local Ollama model can optionally help plan tool calls and phrase answers, which are checked against the facts the tools return; when the model is off or unreachable, the deterministic path is used. Questions are limited to 500 characters. Each answer states whether it came from a verified local model or from deterministic logic, and lists the tools used, any warnings and links to relevant pages. The back end does not persist chat content, and the chat is cleared when the dataset is switched. The assistant is an aid for querying stored data and not an independent investigator.

P. Application Interface

The sidebar links eleven pages: command center, overview, datasets, data quality, theft detection, customer intelligence, feeder analytics, investigations, bill investigation, assistant and settings. A welcome page and routed customer and case pages complete the interface (Figure 6). The top bar offers a command palette, a dataset dropdown showing meters, date range and quality score, and the back-end status. The Settings page holds the acting name, the fixed pipeline constants (feeder z-score 2.0 and risk percentiles 98, 93 and 80), and the API and assistant status. The active dataset is held in client state and resets to the first dataset when the page is reloaded.



Figure 6: GridIntel dashboard showing dataset summary and data-quality indicators.

Q. Deployment

GridIntel targets Windows 10 or 11 with Python 3.11 or newer, Node.js 20 or newer and pnpm, and a modern browser with WebAssembly support for OCR. A launcher script checks the virtual environment, pnpm and free ports and then starts the back end on port 8000 and the front end on port 5173; a companion script stops both.

IV. RESULTS AND DISCUSSION

A. Verification Environment and Scope

A verification run was recorded on 12 November 2026 on Windows 11 with Python 3.12.7, pytest 8.4.2, Node.js 24.21.0 and pnpm 12.10.1, on branch ui-redesign at base commit adbcf58. The raw logs are kept in the project's docs/verification folder. The back-end suite runs against an isolated test database and covers CSV format detection, chunked ingestion and rollback, data-quality checks, synthetic generation and injected patterns, the sixteen features, feeder energy balance, the application programming interface (API) routes, model training and evaluation, the investigation workflow and exports, assistant tools and safety routing, and the stateless bill-comparison endpoint. The results are summarised in Table VII.

Table VII: Summary of the Recorded Software Verification

Check	Recorded result
Backend tests (pytest)	264 collected; 263 passed, 1 skipped; 9 warnings
Backend lint (Ruff)	All checks passed
Backend typecheck (mypy)	No issues in 55 source files
Frontend lint and typecheck	Both exited with code 0
Frontend unit tests	17 passed, 0 failed
Production build	Succeeded in 2.32 s
Backend health endpoint	HTTP 200
Browser smoke test	12 routes; 0 console errors, 0 warnings
Evaluation database	6 datasets, 2,504 customers

Several qualifications apply. The nine back-end warnings are non-fatal; they come from third-party deprecation notices in the test client and from a transaction rollback in test teardown, not from application behaviour. The skipped test is the opt-in live-Ollama smoke test, which runs only when RUN_OLLAMA_SMOKE=1 is set, so the deterministic assistant paths are tested but model-written output is covered only conditionally. The 17 front-end tests cover pure logic in three modules (bill parsing, bill CSV handling and bill review); there is no automated component or browser end-to-end suite, and route rendering was checked manually. The production build emitted a non-fatal bundle-size warning (1,825.22 kB of JavaScript, 573.37 kB gzipped, and 118.73 kB of CSS, 35.50 kB gzipped), which the verification report attributes mainly to the



ECharts and OCR assets. No coverage percentage was measured. The verified state is the working tree of the branch, and the commit identifies its base revision.

B. Observations from the Evaluation Database

A read-only database check found six datasets with 2,504 customers in total, together with a recorded SHA-256 digest of the database file. The selected Residential Grid dataset contained 1,000 meters and a data-quality score of 98.8%. The smoke test opened twelve routes (command center, overview, datasets, data quality, theft detection, customers, feeders, investigations, bill investigation, assistant, settings and welcome) against the running back end, whose health endpoint returned HTTP 200. These observations describe a single snapshot of bundled synthetic or imported evaluation datasets, not a production utility feed. No ROC-AUC, PR-AUC, precision, recall or F1 value from an independent labelled utility dataset was available for this paper. The paper therefore reports no detection performance and makes no claim about real-world accuracy.

C. Discussion

The verification shows that the implemented software builds, passes its automated checks and renders its main routes. It does not show that the models detect theft, that the percentile bands are well calibrated, that the data-quality weights are appropriate or that investigators find the workflow useful. Answering those questions needs labelled utility data and user studies.

The design's main strength is traceability. Features are plain statistical patterns, a missing value is never silently replaced by zero, supervised models are skipped rather than trained on too few labels, and every case keeps the score and reasons that prompted it, along with an append-only audit trail. The interface labels outputs as suspicious, anomalous or investigation leads and avoids stating that theft is confirmed. Because scores are percentile ranks, they show which customers stand out within a cohort, not how likely theft is. The lead-quality summary lets a team see over time whether high-risk leads turned out to be genuine issues, and these recorded outcomes could later provide labels for supervised training.

Several threats to validity should be noted. The software was verified by its developers in a single environment. The datasets are synthetic or imported evaluation sets. The number of front-end tests is small and limited to bill-related logic. The quality weights, risk percentiles and the feeder threshold of 2.0 are fixed constants, not tuned values.

D. Limitations

GridIntel analyses submitted data on request and does not monitor meter streams continuously. The main pipeline needs daily kWh readings, and a few months of data may be too short to tell seasonality, occupancy or load changes, billing-period differences, meter issues, solar changes and suspicious activity apart. A feeder loss combines technical and other losses, and a high score is not proof of theft. Claims about supervised accuracy need ground-truth labels and a suitable held-out dataset, and synthetic results cannot replace validation on real data. The past-bill tool relies on OCR or CSV values that the user must verify. The application has no authentication, so the acting name serves for attribution only, and the audit trail is not tamper-proof.

V. CONCLUSION

This paper presented the design, implementation and software verification of GridIntel, a web-based platform that brings together CSV ingestion, data-quality auditing, sixteen explainable customer features, unsupervised and supervised scoring, cohort-relative risk bands, feeder energy-balance analysis, an audited investigation workflow, a past-bill comparison and a dataset-grounded assistant. Features that cannot be computed remain missing, supervised models are skipped when labels are insufficient, and every score is presented as a lead for human verification. Conclusions about any individual customer must come from checking bills, meters and field conditions.

Future work, which is proposed and is not part of the implemented system, includes validating the platform on real utility data with trustworthy inspection labels and a temporal split; comparing its rankings and bill assessments with the judgements of human reviewers; studying the sensitivity of the quality weights and risk



thresholds; adding authentication so that audit entries can be linked to verified users; and extending the automated tests to components and end-to-end browser behaviour.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering (AI & ML), Jhulelal Institute of Technology, Nagpur, for providing computing infrastructure and institutional support for this work.

DECLARATIONS

Author contributions: A.G. conceived the system architecture, developed the backend analytics pipeline and drafted the manuscript; H.R., S.D., A.B. and S.S. contributed to frontend implementation, testing, verification runs and manuscript review.

Conflict of interest: The authors declare no conflict of interest.

Ethics approval: Not applicable; this study uses synthetic and anonymised evaluation data and involves no human participants.

Data and code availability: The source code, synthetic data generator and verification logs are available from the corresponding author upon reasonable request.

REFERENCES

[1] P. Glauner, J. A. Meira, P. Valtchev, R. State, and F. Bettinger, "The challenge of non-technical loss detection using artificial intelligence: A survey," Int. J. Comput. Intell. Syst., vol. 10, no. 1, pp. 760–775, 2017.

[2] B. Coma-Puig, A. Calvo, J. Carmona, and R. Gavaldà, "A case study of improving a non-technical losses detection system through explainability," Data Min. Knowl. Discov., vol. 38, no. 5, pp. 2704–2732, Sep. 2024.

[3] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," IEEE Trans. Smart Grid, vol. 10, no. 3, pp. 3125–3148, May 2019.

[4] J. Nagi, K. S. Yap, S. K. Tiong, S. K. Ahmed, and M. Mohamad, "Nontechnical loss detection for metered customers in power utility using support vector machines," IEEE Trans. Power Del., vol. 25, no. 2, pp. 1162–1171, Apr. 2010.

[5] J. Nagi, K. S. Yap, S. K. Tiong, S. K. Ahmed, and F. Nagi, "Improving SVM-based nontechnical loss detection in power utility using the fuzzy inference system," IEEE Trans. Power Del., vol. 26, no. 2, pp. 1284–1285, Apr. 2011.

[6] P. Jokar, N. Arianpoo, and V. C. M. Leung, "Electricity theft detection in AMI using customers' consumption patterns," IEEE Trans. Smart Grid, vol. 7, no. 1, pp. 216–226, Jan. 2016.

[7] Z. Zheng, Y. Yang, X. Niu, H.-N. Dai, and Y. Zhou, "Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids," IEEE Trans. Ind. Informat., vol. 14, no. 4, pp. 1606–1615, Apr. 2018.

[8] M. M. Buzau, J. Tejedor-Aguilera, P. Cruz-Romero, and A. Gómez-Expósito, "Detection of non-technical losses using smart meter data and supervised learning," IEEE Trans. Smart Grid, vol. 10, no. 3, pp. 2661–2670, May 2019.

[9] T. V. P. Monteiro, G. J. B. C. Castor, C. G. C. Correa, H. A. C. Arias, D. Z. Ñ. Huatuco, and Y. P. M. Rodriguez, "A hybrid machine learning framework for electricity fraud detection: Integrating Isolation Forest and XGBoost for real-world utility data," Energies, vol. 18, no. 23, Art. no. 6249, 2025.

[10] M. M. Badr, M. I. Ibrahem, H. A. Kholidy, M. M. Fouda, and M. Ismail, "Review of the data-driven methods for electricity fraud detection in smart metering systems," Energies, vol. 16, no. 6, Art. no. 2852, 2023.

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. 8th IEEE Int. Conf. Data Mining, Pisa, Italy, 2008, pp. 413–422.

[12] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[13] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2016, pp. 785–794.



[14] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Proc. Adv. Neural Inf. Process. Syst. 30, 2017, pp. 4765–4774.

[15] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” PLoS ONE, vol. 10, no. 3, Art. no. e0118432, 2015.

[16] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[17] K. Purna Prakash and Y. V. Pavan Kumar, “Simple and effective descriptive analysis of missing data anomalies in smart home energy consumption readings,” J. Energy Syst., vol. 5, no. 3, pp. 199–220, Sep. 2021.