Contents

1 LQS v3.1: A Procurement-Grade Extension to the Quality Intelligence Engine

Authors: LabelSets Research **Contact:** research@labelsets.ai **Date:** April 2026 **Addendum to:** `paper/lqs-v3-paper.md` **Status:** Live in production (labelsets.ai) as of 2026-04-22

1.1 Abstract

We extend the LQS v3 quality intelligence system with seven procurement-grade modules designed to support citation in commercial and regulatory contracts. Where v3.0 answered *"is this dataset high quality?"*, v3.1 answers the harder procurement question: *"can we cite this score in a contract, and will it survive third-party audit?"* Concretely, we add: (1) **Multi-oracle consensus**, which scores a dataset with five architecturally-diverse classical oracles plus two transformer-encoder oracles, reporting coefficient-of-variation–based agreement as a first-class dimension; (2) **Data-driven task inference**, which replaces metadata-declared task routing with an evidence-based taxonomy that also surfaces top-k ambiguity; (3) **Graded contamination scoring**, which converts binary eval-set-leak detection into a 0–100 scale with per-benchmark breakdown and high-stakes weighting; (4) **Dimension- and composite-level 95% confidence intervals** via fold-stdev, Wilson binomial, and bootstrap methods; (5) **Downstream scaling-law projection** that fits `F1 = Fmax − a·n^(-b)` to learning curves and extrapolates F1 at 10×/100× data; (6) **Adversarial label stability**, which measures the rate at which Naive Bayes predictions survive word-drop, char-typo, and numeric-noise perturbations; (7) **Per-demographic subgroup quality**, with heuristic column detection (gender/age/race/region) and disparity scoring; and (8) **Ed25519-signed cryptographic certificates** with a public revocation registry, exposed via `/api/verify-lqs-cert/:hash`. Every cert records the scorer version, oracle registry version, cert version, and a SHA-256 payload hash, making LQS scores independently verifiable by any third party.

Keywords: dataset quality, multi-oracle consensus, confidence intervals, contamination detection, scaling laws, adversarial robustness, fairness, cryptographic certificates, procurement-grade ML

1.2 1. Introduction

LQS v3.0 established dataset-quality scoring as a *predictive* problem: calibrate dimension weights against held-out downstream F1 rather than expert intuition. This is necessary but not sufficient for a procurement use case. A commercial buyer, a regulator, or a ratings analyst needs more:

- **Bias-robustness**: would the score meaningfully change if you used a different oracle? If yes, it reflects the oracle's quirks, not the data.
- **Confidence**: a point estimate of 87 is unciteable without an interval. "LQS = 87 ± 2.1" survives scrutiny.

- **Auditability**: which oracle voted what on which dimension? A contract citation requires an audit trail.
- **Tamper-evidence**: the buyer has to be able to verify that the score they're citing was really produced by the LabelSets scorer, not fabricated.
- **Methodology versioning**: when the scorer methodology updates, existing citations must still resolve to their original semantics.

v3.1 is the layer that delivers those properties. This addendum documents the seven modules we added, the algorithms behind each, the integration into the v3 pipeline, and the deployed API surface.

1.3 2. Multi-Oracle Consensus

1.3.1 2.1 Motivation

Multiple v3.0 dimensions derive from single-oracle predictions: `annotation_consistency` and `label_error_estimate` via 5-fold Naive Bayes, `signal_strength` via a similar procedure. On the empirical side (`benchmark_data`), the Python worker used a single encoder (all-MiniLM-L6-v2) plus LogisticRegression. Single-oracle scoring has a known failure mode: the score inherits the oracle's biases. If MiniLM is weak on retrieval-style tasks, datasets of that type get understated quality signals.

1.3.2 2.2 The Oracle Registry

We introduce `validation/workers/oracle-registry.js` — a declarative registry of every model used as a scoring oracle. Each entry records the oracle's algorithm family, inductive bias, default voting weight, minimum sample requirement, and supported modalities:

```
naive_bayes:            family: 'generative_probabilistic'
knn:                    family: 'distance_based'
decision_tree:          family: 'tree_based'
logistic_regression:    family: 'linear_discriminative'
random_features_nb:     family: 'generative_probabilistic' (random 50% feature subset)
minilm_logreg:          family: 'transformer_embedding' (general pretraining)
bge_logreg:             family: 'transformer_embedding' (retrieval pretraining)
```

The first five run in the Node.js validator (in-memory, per-upload). The last two run asynchronously in the `model-eval` Python worker on Railway, reusing the same Supabase `oracle_evaluations` table as the audit trail. Family diversity is deliberate: five Naive Bayes variants produce weaker agreement signal than NB+KNN+DT+LR+RF, because variants of the same family are correlated in their errors.

1.3.3 2.3 Oracle Agreement as a Dimension

We compute the coefficient of variation of per-oracle held-out accuracy:

$$\mathrm{CV} = \frac{\sigma(\vec{a})}{\mu(\vec{a})}$$

where $\vec{a} = [a_1, \ldots, a_K]$ is the held-out test accuracy of the K registered oracles. We map CV to a 0–100 `oracle_agreement` score via a piecewise-linear curve with break points at CV $\in \{0.02, 0.05, 0.10, 0.20, 0.40\}$. A CV ≤ 0.02 yields 100 (procurement-safe); CV ≥ 0.40 yields 0 (divergent, do not cite a single composite). We also report Fleiss's κ (chance-corrected agreement across all oracles) and the pairwise Cohen κ matrix.

1.3.4 2.4 Audit Trail

Every oracle vote lands in `oracle_evaluations` (Supabase). The schema:

```
oracle_evaluations (
  id uuid PK,
  dataset_id uuid FK,
  scored_at timestamptz,
  scorer_version text,       -- 'lqs_v3.1', 'model_eval_v1.1'
  registry_version text,     -- oracle-registry version snapshot
  dimension text,
  oracle_id text,
  oracle_label text,
  oracle_family text,
  accuracy numeric,
  macro_f1 numeric,
  precision_mean numeric,
  recall_mean numeric,
  support int,
  extra jsonb,               -- pairwise kappa per partner, Fleiss κ group, etc.
  UNIQUE (dataset_id, scored_at, dimension, oracle_id)
)
```

The table is append-only. `REGISTRY_VERSION` is snapshotted into every row, so when the oracle set changes (e.g., adding a new encoder family), historical audit rows remain interpretable.

1.4 3. Data-Driven Task Inference

1.4.1 3.1 The Metadata Failure Mode

v3.0 auto-detected the task profile for task-conditional scoring using `dataset.category` + tags + file format. This is wrong approximately 10–20% of the time in practice. Sellers mislabel (`category: financial_ai` with regression labels; `category: nlp` with tabular data).

1.4.2 3.2 The Inference Procedure

`validation/workers/task-inference.js` infers task from three data-level signals:

1. **Label cardinality**: unique labels / total records.
2. **Label distribution shape**: numeric vs. categorical, float ratio, integer ratio, array ratio.
3. **Record structure**: presence of `instruction/response` fields (LLM fine-tuning), `tokens/ner_tags` parallel arrays (NER), `anchor/positive/negative` triples (retrieval).

For each of 9 task hypotheses, we compute a fit score $\in [0, 1]$:

- `binary_classification`: fit = 1.0 iff cardinality = 2
- `multiclass_classification`: fit = monotone in cardinality up to 50
- `regression`: fit from `numeric_ratio` × `float_ratio` × `unique_ratio`
- `ordinal_regression`: fit when labels are integer-valued within a small range
- `llm_fine_tuning`: detected via alpaca-style / chat / DPO schema
- `ner`, `retrieval_ranking`, `multilabel_classification`, `text_classification`: as documented in code

The top hypothesis drives task-profile selection. When the top-2 hypotheses are within 5 percentage points of each other, the system emits an `ambiguity` flag; the LQS output surfaces both hypotheses to the buyer.

1.4.3 3.3 Disagreement Between Inference and Metadata

When data-driven inference picks task A but metadata says task B, we log `task_disagreement: { data_driven: A, metadata: B }` in the LQS output. This is surfaced in the cert as part of `task_conditional.task_source: 'data_driven'`.

1.5 4. Graded Contamination Scoring

1.5.1 4.1 From Binary to Graded

v3.0 had a binary `is_contamination_clean` flag driven by the fraud pipeline. Procurement requires a graded score: a 5% overlap with MMLU is not the same as 0% or 50%. We added `validation/workers/contamination-scoring.js` which converts the raw scanner output (max similarity, per-benchmark detail) into a 0–100 inverse curve with five tiers (clean / minor / moderate / contaminated / severe).

1.5.2 4.2 High-Stakes Benchmark Penalty

A fixed set of benchmarks (MMLU, GSM8K, HumanEval, HellaSwag, WinoGrande, ARC, TriviaQA, SQuAD, SuperGLUE, BIG-bench, HELM, TruthfulQA, MedQA, PubMedQA, CaseHOLD, LegalBench, …) carry a 5-point penalty because they are widely cited. A 10% overlap with MMLU is qualitatively worse than 10% with an obscure benchmark.

1.5.3 4.3 Procurement Surface

Every LQS cert includes the worst benchmark, top-3 matched benchmarks with similarity rates, tier, and interpretation string. Contract language we've tested: *"LQS contamination_clean ≥ 90 (tier: clean) verified against benchmark fingerprint registry of {n} references."*

1.6 5. Confidence Intervals

1.6.1 5.1 Three Methods

Different dimensions admit different CI methods:

- **Oracle-derived dims** (annotation_consistency, label_error_estimate, signal_strength, oracle_agreement): we reuse the K-fold cross-validation data already produced by `multiModelAgreement`. Per-oracle per-fold accuracies form a natural sampling distribution; stdev / $\sqrt{K}$ is the standard error.
- **Rate-based structural dims** (completeness, uniqueness, format_integrity, schema_validity): Wilson binomial CI on the underlying proportion. Robust to extreme rates (0% and 100%) and to small sample sizes.
- **Composite**: bootstrap with k=200 iterations over dim-level scores, sampling each dim from $\mathcal{N}(\hat{s}, \hat{\sigma})$ where $\hat{\sigma}$ is the dim's SE when known, else zero.

1.6.2 5.2 Symmetric Clamping

A technical subtlety: `computeLQSv3` post-hoc blends the general-statistical composite with empirical and adversarial scores. The resulting `finalScore` can differ from the mean of the bootstrap distribution. To guarantee the CI encloses the point estimate, we express the CI as $[\hat{s}-1.96\hat{\sigma}, \hat{s}+1.96\hat{\sigma}]$ with $\hat{\sigma}$ from the bootstrap. Percentile bounds from the raw samples are still reported alongside (`bootstrap_p2_5`, `bootstrap_p97_5`) for full transparency.

1.6.3 5.3 Procurement Citation Form

"LQS = 87 ± 2.1 (95% CI: [84.9, 89.1])"

1.7 6. Downstream Scaling-Law Projection

1.7.1 6.1 The Buyer Question

"If I train my model on 10× this data, what F1 do I get?" Direct answer requires a scaling-law fit. v3.0 already had `dataEfficiencyCurve`, which trains the inline Naive Bayes at 10/25/50/75/100% fractions and measures held-out F1. We extended this with `validation/workers/downstream-projection.js`, which fits the classical saturating scaling law

$$F_1(n) = F_{\max} - a \cdot n^{-b}$$

via log-linearized least squares, searching over `Fmax` ∈ `{0.90, 0.92, ..., 1.00}` for the value that maximizes R^2. Once fitted, we extrapolate to 10× and 100× the current sample count, and invert the law to report n at which $F_1 = 0.95F_{\max}$ (the "95%-of-ceiling" point).

1.7.2 6.2 Efficiency Tiers

We categorize datasets into four tiers based on fit parameters:

- **saturated**: $F_{\max} - F_1(n_{\text{cur}}) < 0.015$
- **plateauing**: gap in [0.015, 0.05]
- **learning**: gap in [0.05, 0.15]
- **data_hungry**: gap > 0.15

We derive `downstream_headroom` 0–100 as $\min(100, \max(0, \text{gap}/0.15 \times 100))$, scaled so a 15pp gap is full headroom.

1.7.3 6.3 Interpretation

Saturated data is *more* valuable than data-hungry data at the same current F1: no return on additional purchase. But the headroom score is not a quality penalty — it's a *buyer decision signal*. It carries a low 1% composite weight to reflect this.

1.8 7. Adversarial Label Stability

1.8.1 7.1 Motivation

Labels that flip under small perturbations are fragile. A dataset where 5% word noise produces 40% prediction changes will train a model that doesn't generalize to real-world input drift. v3.0 measured adversarial *robustness* (ARP) by flipping label subsets and observing degradation curves. v3.1 measures *input-side* stability: perturb the inputs, not the labels, and check prediction stability.

1.8.2 7.2 Perturbations

For NLP: `word_drop` (20% tokens dropped), `char_typo` (10% char mutations — swap/delete/duplicate), `word_shuffle` (15% adjacent-pair swaps). For tabular: `gaussian_noise` ($\sigma = 0.10 \times$ feature stdev), `feature_drop` (10% of numeric features replaced with column mean).

1.8.3 7.3 Stability Score

For each sample, we apply $K = 3$ perturbations, run the trained Naive Bayes, and record the fraction of perturbations whose predicted class matches the original. Overall stability is the mean over all samples; we also report per-class stability and per-perturbation-kind stability. Tiers: `rock_solid` / `stable` / `moderate` / `fragile` / `chaotic`.

1.9 8. Per-Demographic Subgroup Quality

1.9.1 8.1 Regulatory Reality

Equal-treatment regulation in finance (ECOA, Fair Lending), healthcare (HHS 1557), and employment (EEOC) requires dataset-level evidence that training data is not systematically skewed against protected subgroups. LQS v3.1 detects demographic columns by two heuristics:

1. **Column-name match**: regex over known demographic names (gender, sex, age, age_bucket, race, ethnicity, country, region, state, language, education, income, marital_status).
2. **Value-shape inference**: columns whose values match known demographic patterns (`{M, F}`, `{male, female}`, age bucket patterns like `18-24`).

1.9.2 8.2 Disparity Metrics

For each detected column with ≥ 2 subgroups, we compute:

- `subgroup_count`
- `min_subgroup_support`, `max_subgroup_support`
- `imbalance_ratio = max / min`

- `class_distribution_disparity`: maximum range of reference-class prevalence across subgroups

1.9.3 8.3 Equity Score

We penalize: small subgroups ($< 30 \rightarrow$ -35, 30–99 $\rightarrow$ -10); imbalance ratio ($> 20 \rightarrow$ -20, 5–20 $\rightarrow$ -5); class prevalence disparity ($> 0.30 \rightarrow$ -30, 0.15–0.30 $\rightarrow$ -15, 0.05–0.15 $\rightarrow$ -5). Final equity = worst column's score.

1.9.4 8.4 Absence Doctrine

If no demographic columns are detected, we do **not** score the dimension (dim omitted from composite). Datasets legitimately lack demographic signal for many ML tasks — scoring them at 50 neutral would penalize honest omission and reward seller bias (adding fake demographic columns to inflate the score).

1.10 9. Cryptographically Signed Certs

1.10.1 9.1 Cert Schema

Every LQS v3.1 run issues an Ed25519-signed certificate containing:

- Issuer, methodology URL (including version anchor)
- Dataset ID, scored_at, scorer_version
- Composite + composite_ci_95 (low/high/SE)
- Tier, confidence, verified flag
- Task-conditional (task, task_source, composite)
- Oracle consensus (registry_version, agreement_score, tier, oracles_counted, kappa_fleiss)
- Contamination (version, score, tier, worst_benchmark, benchmarks_checked)
- Downstream projection (version, tier, headroom_score, scaling_law coefficients + R^2)

1.10.2 9.2 Canonical Serialization + Signing

We canonicalize JSON by sorting object keys alphabetically, minifying, and encoding as UTF-8. The canonical byte string is signed with Ed25519 (node:crypto.sign with `null` algorithm — the Ed25519 default). The cert includes:

- `signature_b64`: base64-encoded 64-byte Ed25519 signature
- `cert_hash`: SHA-256 of the canonical payload (identifier for verification + revocation)
- `public_key_id`: first 16 chars of SHA-256 of the public key DER encoding (key rotation support)

Canonical JSON serialization lives in both `validation/workers/lqs-cert.js` (ESM, signer side) and `api/lqs-cert-verify.js` (CommonJS, verifier side). The implementations are kept in sync by documentation; a drift between them would break all cert verification.

1.10.3 9.3 Public Verification Endpoint

`GET /api/verify-lqs-cert/:cert_hash` is unauthenticated — it is the procurement verification surface. It returns:

```
{
  "valid": true|false,
  "status": "valid"|"revoked"|"tampered"|"not_found"|"unverified",
  "cert": { payload, signature_b64, cert_hash, public_key_id, key_source, issued_at, ... },
  "revocation": null | { revoked_at, reason, reason_code },
  "verification": { valid, reason, cert_hash }
}
```

`GET /api/lqs-public-key` returns the current Ed25519 public key in PEM format, enabling offline verification by third parties.

1.10.4 9.4 Revocation Registry

`lqs_cert_revocations` is append-only. If post-facto analysis finds contamination, PII leak, or other procurement-material issue, we revoke the cert. The revocation adds a row; the original cert stays in `lqs_certs` (immutable history). A verifier that queries a revoked cert gets `status: "revoked"` with the revocation reason and timestamp.

1.11 10. Live Evidence of Procurement-Grade Behavior

1.11.1 10.1 Deployment Summary (2026-04-22)

- Validation service (Railway): `lqs-v3.js` v3.1 + all seven modules deployed. Signing key `aa4c070af907e2ea` stored as `LQS_SIGNING_PRIVATE_KEY` env var.
- API service (Railway): `/api/verify-lqs-cert/:hash`, `/api/lqs-public-key` live.
- Model-eval service (Railway): Python worker runs MiniLM + BGE dual-oracle empirical evaluation; audit rows land in `oracle_evaluations`.

1.11.2 10.2 Verified End-to-End Example

- Signed cert `27a8060ab73c6d9c…` for dataset `ebe064ba-c411-467f-83f5-6ba979dd917f` produced locally
- Inserted into `lqs_certs` via service_role
- Verified via `GET https://initialdeploy-production.up.railway.app/api/verify-lqs-cert/27a8060ab73c... → {valid: true, status: "valid"}`
- Public key fingerprint in cert matches the published fingerprint

1.11.3 10.3 Observed Multi-Oracle Results (marketplace samples)

Two NLP datasets processed through the Phase 1 Python multi-oracle path produced:

Dataset	agreement_score	CV	sample_agreement	kappa_fleiss	tier
`d9e75396`	100	0.0056	0.387	—	consensus
`37ec70ac`	100	0.000	1.000	—	consensus

Dataset `d9e75396` is especially informative: *accuracies* across MiniLM and BGE agree tightly (CV 0.006), but the *per-sample predictions* disagree (sample_agreement 39%). This is the ideal signal for an ensemble — diverse errors on individual samples, identical aggregate performance.

1.12 11. Reproducibility and Standards

1.12.1 11.1 Published Materials

All modules are available in the LabelSets repository: - `validation/workers/oracle-registry.js` - `validation/workers/task-inference.js` - `validation/workers/contamination-scoring.js` - `validation/workers/confidence-intervals.js` - `validation/workers/downstream-projection.js` - `validation/workers/adversarial-stability.js` - `validation/workers/subgroup-quality.js` - `validation/workers/lqs-cert.js`

Each module includes a version constant. The cert payload includes every version: `scorer_version`, `registry_version`, `contamination_scoring_version`, etc. A future auditor can always determine which exact algorithm produced a cited LQS.

1.12.2 11.2 Smoke-Test Suite

Eight independent smoke-test files (`smoke-test-*.mjs`) cover ~140 assertions exercising happy path, edge cases, degenerate inputs, and integration boundaries for every module. All pass in sequence; a pre-merge regression script runs all eight.

1.12.3 11.3 Standards Track

v3.1 is designed to be submitted to a standards body (IEEE P2841-style dataset quality standard, or NIST AI RMF alignment) with the canonical-JSON + Ed25519 scheme serving as the reference implementation. We invite external audits of the methodology, including the specific piecewise-linear curves for oracle agreement and contamination scoring.

1.13 12. Conclusion

LQS v3.0 answered a predictive question. v3.1 answers a **citation** question. The procurement-grade extensions — multi-oracle consensus, graded contamination, confidence intervals, scaling-law projection, adversarial stability, subgroup quality, and cryptographic certs — turn a useful quality score into a trust mark that can survive commercial, regulatory, and standards-body scrutiny. The 104+ deployed assertions, the public verification endpoint, and the signed-cert protocol constitute the deliverable: *"LQS Verified v3.1"* as a citable trust mark, analogous to an ISO certification or credit-ratings-agency grade.

1.14 Appendix A: CALIBRATED_WEIGHTS (v3.1, 19 dimensions)

Dim	Base weight	Source
annotation_consistency	0.292	v3.0 calibration
class_distribution	0.268	v3.0 calibration
label_error_estimate	0.263	v3.0 calibration
uniqueness	0.040	v3.0 calibration

Dim	Base weight	Source
distribution_health	0.027	v3.0 calibration
label_accuracy	0.027	v3.0 calibration
signal_strength	0.017	v3.0 calibration
size_adequacy	0.012	v3.0 calibration
completeness	0.010	hygiene floor
schema_validity	0.010	hygiene floor
format_integrity	0.010	hygiene floor
label_density	0.010	hygiene floor
diversity_score	0.010	hygiene floor
provenance_quality	0.010	hygiene floor
oracle_agreement	**0.020**	**v3.1 Phase 1**
contamination_clean	**0.020**	**v3.1 Phase 3**
downstream_headroom	**0.010**	**v3.1 Phase 5**
adversarial_stability	**0.020**	**v3.1 Phase 6**
subgroup_equity	**0.020**	**v3.1 Phase 7 (conditional)**

1.15 Appendix B: High-stakes Task Profile Multipliers (v3.1 additions)

Profile	oracle_agreement	contamination_clean	adversarial_stability	subgroup_equity
legal_ai	3.0	2.5	2.0	3.0
medical_ai	3.5	2.5	2.5	4.0
financial_ai	2.5	2.0	3.0	3.5
llm_fine_tuning	2.0	3.5	—	—

1.16 Appendix C: Cert JSON Shape (canonical form, sorted keys)


```
{
  "composite": 87,
  "composite_ci_95": { "high": 88.8, "low": 85.2, "se": 0.9 },
  "confidence": "high",
  "contamination": {
    "benchmarks_checked": 5,
    "score": 95,
    "tier": "clean",
    "version": "1.0",
    "worst_benchmark": "MMLU"
  },
  "dataset_id": "d9e75396-2629-44dc-a609-4280d5571038",
  "downstream_projection": {
    "headroom_score": 45,
    "scaling_law": { "Fmax": 0.95, "a": 0.6, "b": 0.3, "r_squared": 0.95 },
    "tier": "learning",
    "version": "1.0"
  },
```

```
  "issuer": "Labelsets LLC",
  "issuer_id": "lqs-labelsets-v3.1",
  "methodology_url": "https://labelsets.ai/lqs-methodology#v3.1",
  "oracle_consensus": {
    "agreement_score": 92,
    "kappa_fleiss": 0.84,
    "oracles_counted": 5,
    "registry_version": "1.1",
    "tier": "consensus"
  },
  "scored_at": "2026-04-23T00:59:45.219Z",
  "scorer_version": "3.1",
  "task_conditional": {
    "composite": 86,
    "task": "binary_classification",
    "task_source": "data_driven"
  },
  "tier": "gold",
  "v": "1.0",
  "verified": true
}
```