International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 04 April-2026 | Impact Factor: 3.5

A Framework for Rigorous and Replication ML Model Assessment: Integrating Statistical Significance and Practical Evaluation

Rakshit Ranjan Singh[1], Adlin Jebakumari S[2], Avani Singh[3], Kazim Mahadi[4]

[1,3,4]Student, School of Computer Science and Information Technology,
JAIN (Deemed-to-be University), Bengaluru, India

[2]Assistant Professor, School of Computer Science and Information Technology,
JAIN (Deemed-to-be University), Bengaluru, India

How to Cite this Article:

Singh, R. R., Singh, A. & Mahadi, K. (2026). A Framework for Rigorous and Replication ML Model Assessment: Integrating Statistical Significance and Practical Evaluation. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.234

https://doi.org/10.55041/ijcope.v2i4.234

Abstract

The rapid study of machine learning (ML) across major infrastructure, from healthcare diagnostics to financial risk assessment, has brought serious problems in model evaluation. A common problem in the field is the "accuracy trap." This happens when models are considered better based on accuracy level without assessing whether these gains are statistically genuine or practically meaningful. Furthermore, the field is struggling with a "replication problem," where the lack of versioned data, code, and environment details makes it difficult to verify many performance. This paper proposes a solution in the form of a Dual-Pillar Validation Framework. We introduce a methodology that combines strong statistical hypothesis testing (specifically Bootstrap Confidence Intervals and the 5x2cv paired t-test) with practical evaluation audits (covering fairness, data drift, and latency). By developing this framework within a repeatable environment MLOps pipeline utilizing Docker, DVC, and MLflow, we establish an automated "guardian" mechanism. This ensures that deployed models are not performing by coincidence, but are statistically strong, ethically sound, and efficient in practice.

Keywords— Machine Learning Evaluation, Statistical Validation, Reproducibility, MLOps

1. Introduction

Machine Learning (ML) has moved beyond theory and become the core or backbone of modern software systems. Still, the validating ML model often lags behind traditional software testing standards. A common, yet dangerous, anti-pattern in the industry involves selecting a "champion" model based solely on a trivial increase in a metric like accuracy or F1-score—for instance, preferring a model with 94.5% accuracy over one with 94.2%—without investigating the stability of that result.[1] This "shallow evaluation" fails to distinguish between genuine algorithmic advancement and the random noise inherent in dataset splitting and stochastic initialization.

The consequences of deploying models based on weak evidence can be serious. A model that appears barely better on a static test set may fail to remain accurate outside the training environment, show strong bias against protected groups, or add excessive delay that degrades user experience. Adding to the problem is the "replication problem " highlighted by Pineau et al. (2021), which shows that most of the portion of ML research cannot be reproduced due to missing hyperparameters, unshared code, or undocumented random seeds often missing or not shared.[2]

This research aims to formalize a **Dual-Pillar Validation Framework** to address these shortcomings:

- **Pillar 1 (Statistical Rigor):** This pillar answers the question, *"Is the improvement real?"* It uses careful statistical tests to make sure that results aren't just due to random chance.
- **Pillar 2 (Practical Significance):** This pillar answers the question, *"Does the improvement matter?"* It looks at the size of effect while also considering important factors like fairness, inference speed, and robustness to data drift.

By building these pillars into a Continuous Integration/Continuous Delivery (CI/CD) pipeline, we shift from one–off analysis to a systematic, automated engineering process [4]

2. Literature Review

The framework is built on a foundation that combines statistical inference, software engineering (MLOps), and principled ethical AI.

2.1 The Incumbent of Replication

Replication is the cornerstone of scientific research. In ML, it implies that an independent user can achieve the same results using the same data, code, and settings. However, "inherent nondeterminism" in GPU operations and software library dependencies often leads to fluctuating performance, making comparisons difficult.[5] Tools like **Docker** have become essential for encapsulating the runtime environment [6], while **Data Version Control (DVC)** has emerged as the standard for tracking data lineage and pipeline stages, ensuring that the exact dataset used for testing can be retrieved years later.[7] Without these tools, any claim of statistical significance is moot because the experiment itself is unstable.

2.2 Statistical vs. Practical Significance

A critical distinction often overlooked in ML evaluation is the difference between statistical and practical significance. Statistical significance, checked by p-values, indicates the probability that an observed difference occurred by random chance.[9] However, Lin et al.(2013) warn that with the big datasets common in ML, even insignificant, irrelevant differences can yield statistically significant p-values.[10]

Opposite, practical significance refers to the magnitude of the effect - whether the gain is large enough to justify the cost of deployment.[11] As noted in recent statistical reviews, assurance on p-values alone encourages "p-hacking," where researchers mine data for any "significant" result while ignoring the practical applicability.[12] Confidence intervals (CIs) are increasingly recommended over simple p-values because they quantify the size of the improvement and the confusion surrounding it.[14]

2.3 Methodologies for Statistical Comparison

Standard parametric tests, such as the Student's t-test, are frequently blown in ML. Dietterich (1998) demonstrated that standard k-fold cross-validation violates the self-reliance assumption required by t-tests, leading to high Type I error rates (false positives). [15]

- **5x2cv Paired t-test:** To correct for this, Dietterich proposed replicating 2-fold cross-validation five times, a method now supported by libraries like mlxtend.[15]

- **McNemar's Test:** For large deep learning models where retraining is computationally excessive, McNemar's test is the preferred non-parametric method. It operates on a single test set's probability table, analyzing where models disagree. [17]

- **Bootstrap Resampling:** This non-parametric path involves resampling the test set with replacement to generate a distribution of performance differences. It highly soft and allows for the direct calculation of confidence Intervals, making it a gold standard for modern evaluation. [19]

2.4 Auditing for Practical Readiness

- Modern MLOps principles dictate that accuracy is necessary, but faulty, condition for deployment.

- **Data Drift:** Production data surely diverges from training data. Tools like Evidently AI utilize statistical distance metrics (e.g, Kolmogorov-Smirnov, Wasserstein distance) to detect these shifts early. [21]

- **Fairness:** Algorithmic bias is a critical risk. Frameworks like IBM's AI Fairness 360 (AIF360) provide Uniform metrics such as Disparate Impact and Equalized Odds to compute bias against protected groups. [23]

- **Latency:** Finally, the computational cost of the assumption is a key practical metric. For Large Language Models (LLMs), metrics like "Time To First Token" (TTFT) are crucial to determine user experience.[25]

3. Materials and Methods

This study employs a constructive research design, developing a conceptual framework and technical architecture. The proposed framework is designed to function as an automated " Gatekeeper" within a CI/CD pipeline.

3.1 Framework Architecture

The Validation pipeline is coordinated using DVC to manage the dependency graph (Prepare -> train -> evaluate).[7] The entire execution occurs within Docker containers to guarantee environmental reproducibility.[6] All metrics and artifacts are logged to ML flow, which serves as the centralized system of record. [28]

3.2 Pillar 1: Statistical Validation Methodology

To assess the reality of a performance gain, we utilize **Non-Parametric Bootstrap Resampling**. This method was selected for its ability

y to generate Confidence Intervals without assuming a normal distribution of errors.[19]

Procedure:

1. **Resampling:** From the selected test set of size N, we draw B = 1000 bootstrapped samples.

2. **Evaluation:** For every sample of b, we calculate the performance difference by applying this formula:

$\delta_b = \text{Metric}_{\text{candidate}} - \text{Metric}_{\text{production}}$

3. **Interval Estimation:** We detect the 95% Confidence Interval (CI) from the distribution of δ by applying the percentile method.[14]

4. **Decision Rule:** If the Confidence Interval strictly excludes zero (e.g., [0.01, 0.03]), the finding is considered statistically significant.

3.3 Pillar 2: Practical Fitness Methodology

A Machine Learning model that passes the statistical test must then pass a list of practical tests:

- **Drift Detection:** We use the **Evidently AI** library to run a Kolmogorov-Smirnov test on quantitative variables. If the amount of drifted features overreaches a pre-set limit (e.g., 35%), the model is marked as vulnerable.[21]

- **Fairness Audit:** We use the **AIF360** toolkit to calculate the *Disparate Impact* ratio. The pipeline applies a "fairness constraint" where the ratio should be between 0.8 and 1.25.[23]

- **Computational Cost:** We benchmark **P95 Inference Latency** inside the production container. The build fails if the latency goes beyond the Service Level Agreement (SLA).[25]

3.4 The "Gatekeeper" Integration

The final module is an automated Python script that queries the MLflow tracking server. It fetches the CI bounds, drift scores, and bias metrics. It applies rigid business logic—requiring *both* statistical significance

and practical compliance—to conclusively Pass or Fail the CI/CD build.[4]

4. Discussion

4.1 Mitigating P-Hacking through Pre-Registration

The multiple comparisons problem (or p-hacking) is a universal concern in statistical testing. If a researcher runs enough reconfigurations of a model, random chance shows they will ultimately discover a "significant" result.[12] This framework reduces that risk through **code-based pre-registration**. By hard-coding the evaluation metrics, failure thresholds, and comparison baselines into the dvc.yaml pipeline configuration *before* the experiment is executed, we implement a disciplined testing protocol that helps avoid post-hoc cherry-picking.[36]

4.2 The Role of Domain Expertise

While this framework automates the rejection of weak models, human verdicts stay vital for defining "practical significance." A 0.15% accuracy gain might be insignificant in ad targeting but critical in diverse medical treatments.[37] Therefore, the boundaries used in the Gatekeeper script are not global constants; subject matter experts must fine-tune them. Moreover, the framework generates interpretability artifacts, such as Shapley Additive exPlanations plots, which allow human reviewers to examine *why* a model is making decisions. This assures that the model depends upon valid causal features rather than misleading correlations.[38]

5. Conclusion

The current industry standard for Machine learning model evaluation is inadequate for high-stakes deployment. By ing on simple, single-point accuracy comparisons, professional risk deploying models that are statistically unstable, ethically biased, or operationally expensive. This paper has presented a rigorous, dual-pillar framework that integrates **statistical significance testing** with **practical auditing**. By implementing this framework with reproducible MLOps tools such as Docker, DVC, and MLflow, organizations can transition from ad hoc model selection to a disciplined, engineering-grade validation process. This ensures that AI systems are not just accurate on paper, but reliable, fair, and robust in the real world.

6. References

1. Raschka, S. (2018). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv preprint arXiv:1811.12808*.

2. Pineau, J., et al. (2021). Improving reproducibility in machine learning research (A report from the NeurIPS 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164), 1-20.

3. Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9).

4. Google Cloud Architecture Center. (n.d.). MLOps: Continuous delivery and automation pipelines in machine learning. Retrieved from Google Cloud Documentation.

5. Nagarajan, V., et al. (2019). Deterministic Implementations for Reproducibility in Deep Learning. *NeurIPS Workshop*.

6. Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1), 71-79.

7. Kuprieiev, R., et al. (2021). DVC: Data Version Control - Git for Data & Models. *Iterative*.

8. Barrak, A., et al. (2021). Analysis of Data Versioning Tools for Machine Learning Operations. *Medium*.

9. Scribbr. (2022). Statistical significance vs. practical significance. Retrieved from Scribbr.com.

10. Lin, M., Lucas, H. C., & Shmueli, G. (2013). Too Big to Fail: Large Samples and the p-Value Problem. *Information Systems Research*, 24(4), 906-917.

11. Statistics By Jim. (2022). Practical vs. Statistical Significance. Retrieved from StatisticsByJim.com.

12. Head, M. L., et al. (2015). The Extent and Consequences of P-Hacking in Science. *PLOS Biology*, 13(3).

13. Wasserstein, R. L., & Lazar, N. A. (2016). The ASA Statement on p-Values: Context, Process, and Purpose. *The American Statistician*, 70(2), 129-133.

14.Raschka, S. (2022). Confidence Intervals for Machine Learning. Retrieved from SebastianRaschka.com.

15.Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7), 1895-1923.

16.Raschka, S. (2020). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *Journal of Open Source Software*.

17.McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12, 153–157.

18.Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7, 1-30.

19.Efron, B., & Tibshirani, R. J. (1994). An Introduction to the Bootstrap. *CRC Press*.

20.Brownlee, J. (2018). A Gentle Introduction to the Bootstrap Method. *Machine Learning Mastery*.

21.Evidently AI. (2021). Data Drift Detection for Machine Learning. Retrieved from EvidentlyAI.com.

22.Lu, J., et al. (2018). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*.

23.Bellamy, R. K. E., et al. (2018). AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. *arXiv preprint arXiv:1810.01943*.

24.Mehrabi, N., et al. (2021). A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys*, 54(6).

25.Databricks. (2023). LLM Inference Performance Engineering Best Practices. Retrieved from Databricks Blog.

26.Hermann, J. (2018). Reproducible Machine Learning with DVC. *SysML Conference*.

27.Kreuzberger, D., et al. (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*.

28.Zaharia, M., et al. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41(4), 39-45.