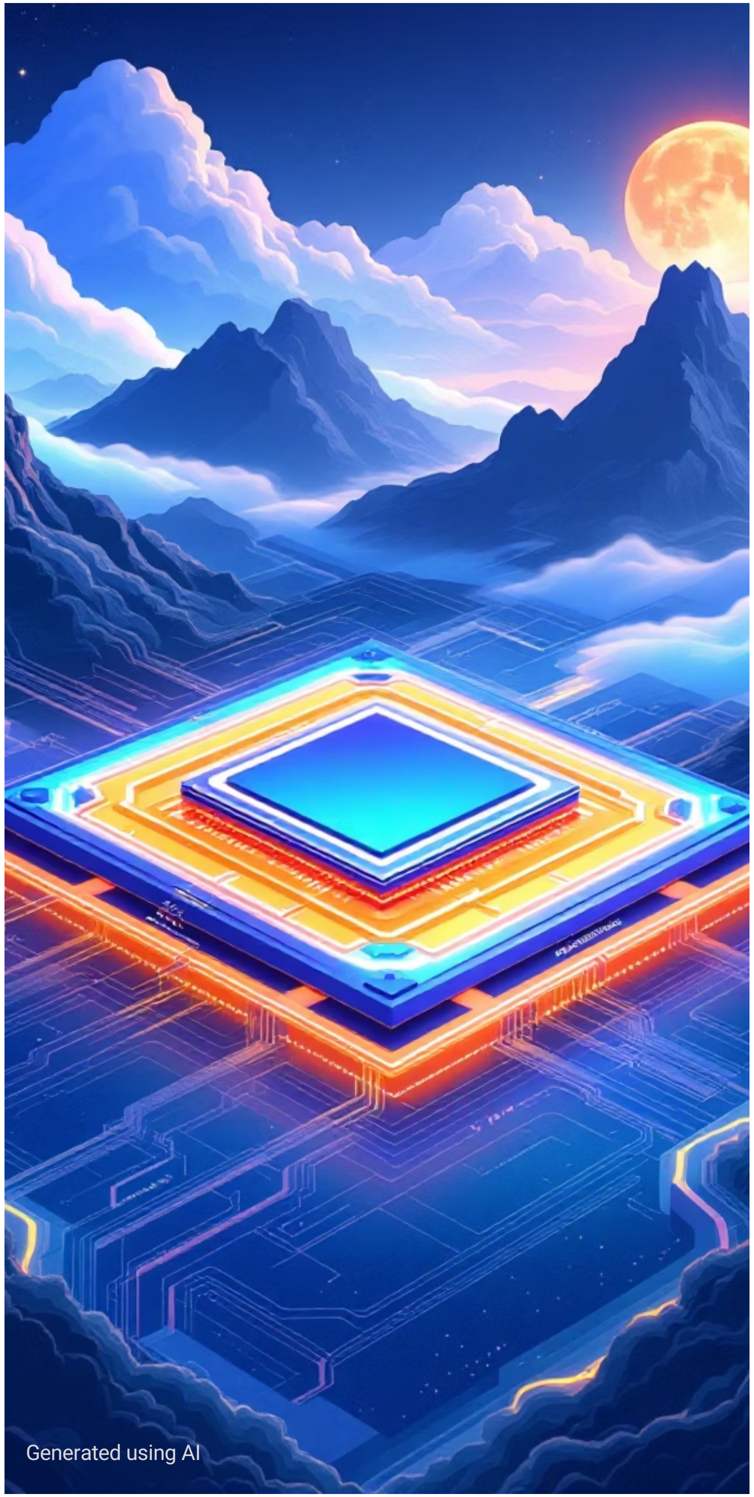
Evaluating Apple's MLX Framework for Machine Learning

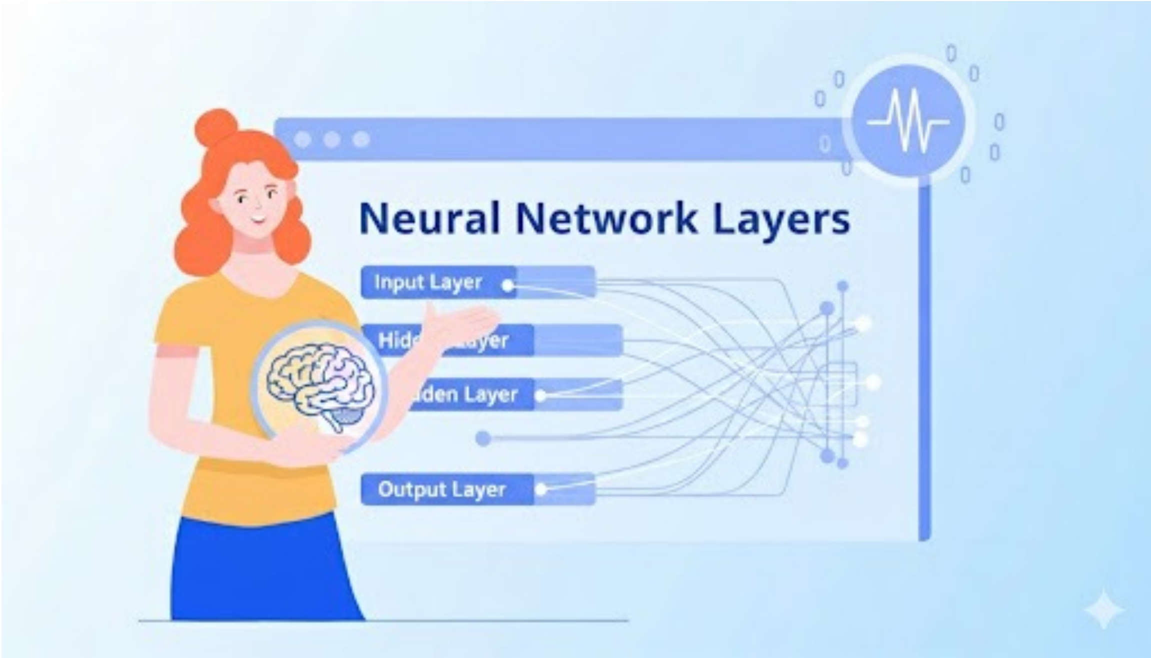
A Comparative Study Using Unsupervised Clustering Algorithms

Sharan Deepak Thakur
GH1031360



Agenda

1. Introduction
2. Literature Review
3. The Research Gap
4. Research Questions
5. Methodology & Conceptual Framework
6. Experimental Methodology
7. Challenges
8. Results
9. Conclusion



Generated using AI

Introduction

Machine Learning

Supervised Learning

Unsupervised Learning

Clustering

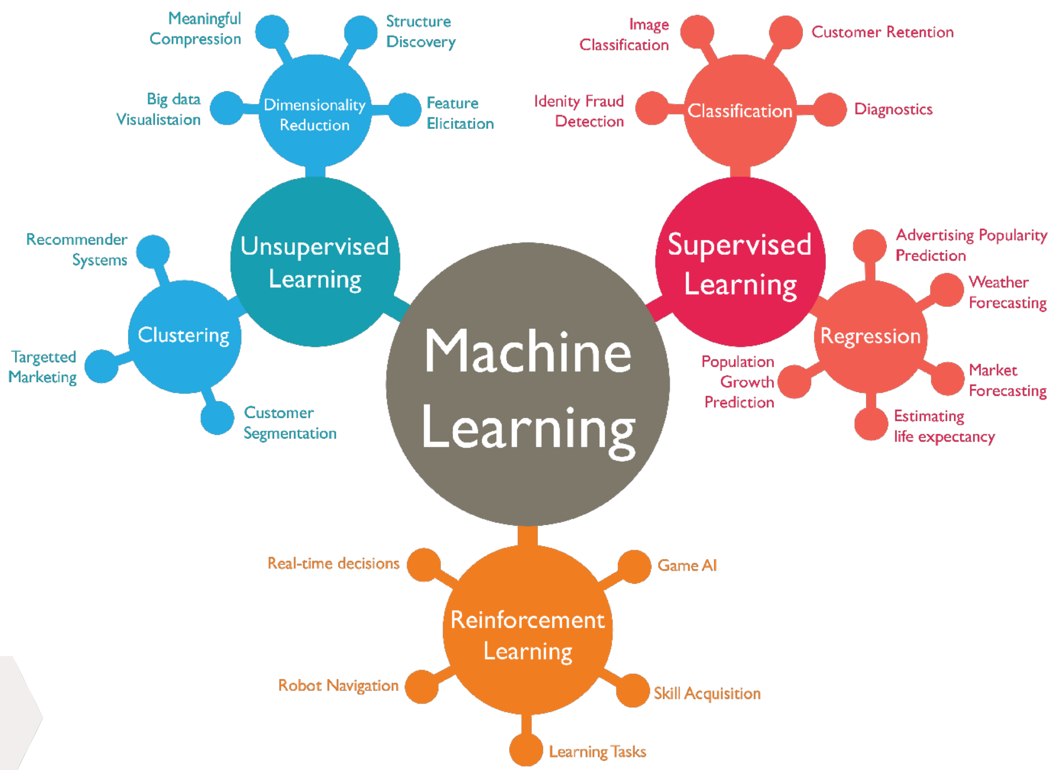


Fig: Machine Learning Problems
Dan Shewan WordStream, 2023

The Evolution of Machine Learning Hardware

1

GPU-Centric Era

Traditional deep learning frameworks optimized for NVIDIA CUDA architecture with discrete GPU memory.

Apple Silicon Shift

1. M-series processors
2. Unified memory architecture integrating CPU, GPU, ANE with shared pools up to 192GB.

MLX Framework

Purpose-built array framework designed specifically to leverage Apple Silicon's unique architecture from the ground up.

Literature Review

Paper Author	Relevance
Awni Hannun et al., 2025	Defines the MLX framework's architecture, optimizations, and rationale for Apple Silicon benchmarking.
Indranil Bose & Xi Chen, 2009	It formalizes both hybridization strategies (cluster → classifier and cluster label as feature).
Hoang Tran, Ngoc Le, Van-Ho Nguyen, 2023	Uses K-Means for segmentation on financial data (credit card/banking).

Table: Relevant Papers

KMeans Clustering:

Hartigan and Wong. 1979, Most widely deployed unsupervised algorithm for customer segmentation, financial analytics, and exploratory analysis

MLX Design Philosophy:

Hannun et al. 2025 designed MLX with lazy evaluation and unified memory utilization

Apple Silicon Architecture:

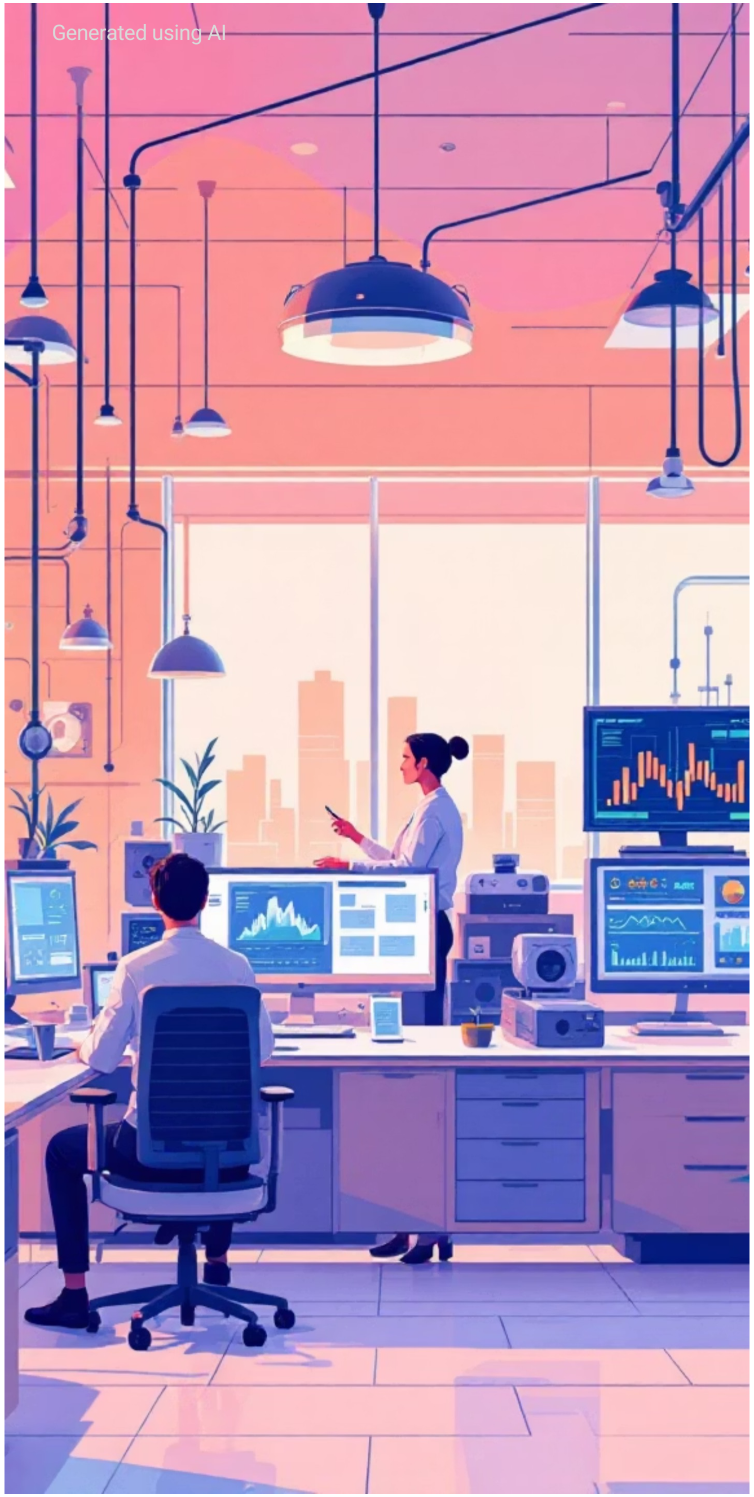
Feng et al. 2025 reveals comprehensive profiling unified memory eliminates CPU-GPU data transfer overhead

The Research Gap

Current literature overwhelmingly focuses on MLX performance for deep learning specifically LLMs and transformers. However, **a critical gap exists for traditional machine learning algorithms**.

Algorithmic Diversity

Production Relevance



Research Questions

1

O1. Performance Comparison

RQ1. *How does MLX's computational performance for KMeans clustering compare to scikit-learn, NumPy, and PyTorch on Apple Silicon?*

2

O2. Clustering Quality

RQ2. *Does MLX produce equivalent results measured by Adjusted Rand Index, Normalized Mutual Information, and Silhouette Score?*

3

O3. API Usability

RQ3. *What are the primary implementation challenges when developing traditional ML algorithms in MLX given its API constraints?*

4

O4. Practical Implications

RQ4. *What actionable recommendations should practitioners consider when evaluating MLX for unsupervised learning workflows?*

Methodology & Conceptual Framework

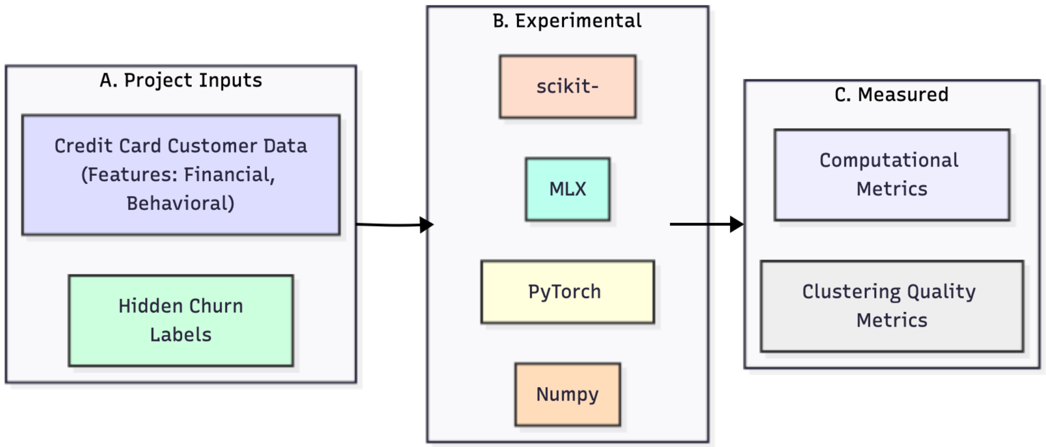


Fig: Proposed Methodology for experiment.

Made with mermaidchart.com

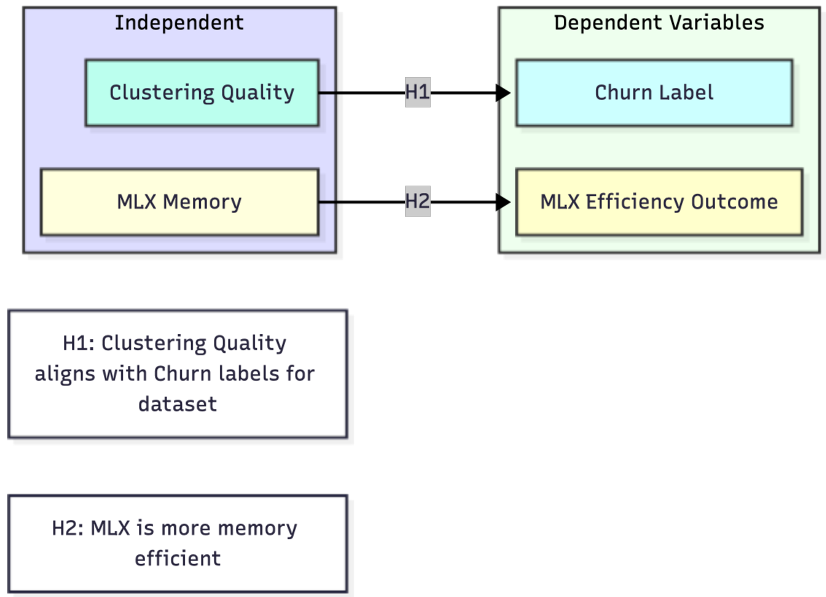


Fig: Conceptual Framework

Hypotheses

- **H1:** Clustering Quality → Label
  - Is the clustering quality aligned with the churn labels of the labelled data?
- **H2:** Memory Efficiency
  - Does MLX use less memory than other frameworks.

Experimental Methodology

01

Multi-Framework Implementations

Parallel implementations of KMeans across MLX, scikit-learn, NumPy, and PyTorch using identical algorithmic logic and parameters.

02

Rigorous Benchmarking Protocol

Experimental rounds following Bahrampour et al. from 2016's framework comparison methodology.

03

Controlled Environment

All experiments executed on identical Apple Silicon hardware, Python 3.11, with standardized data preprocessing and feature scaling between 0-1.

04

Comprehensive Metrics

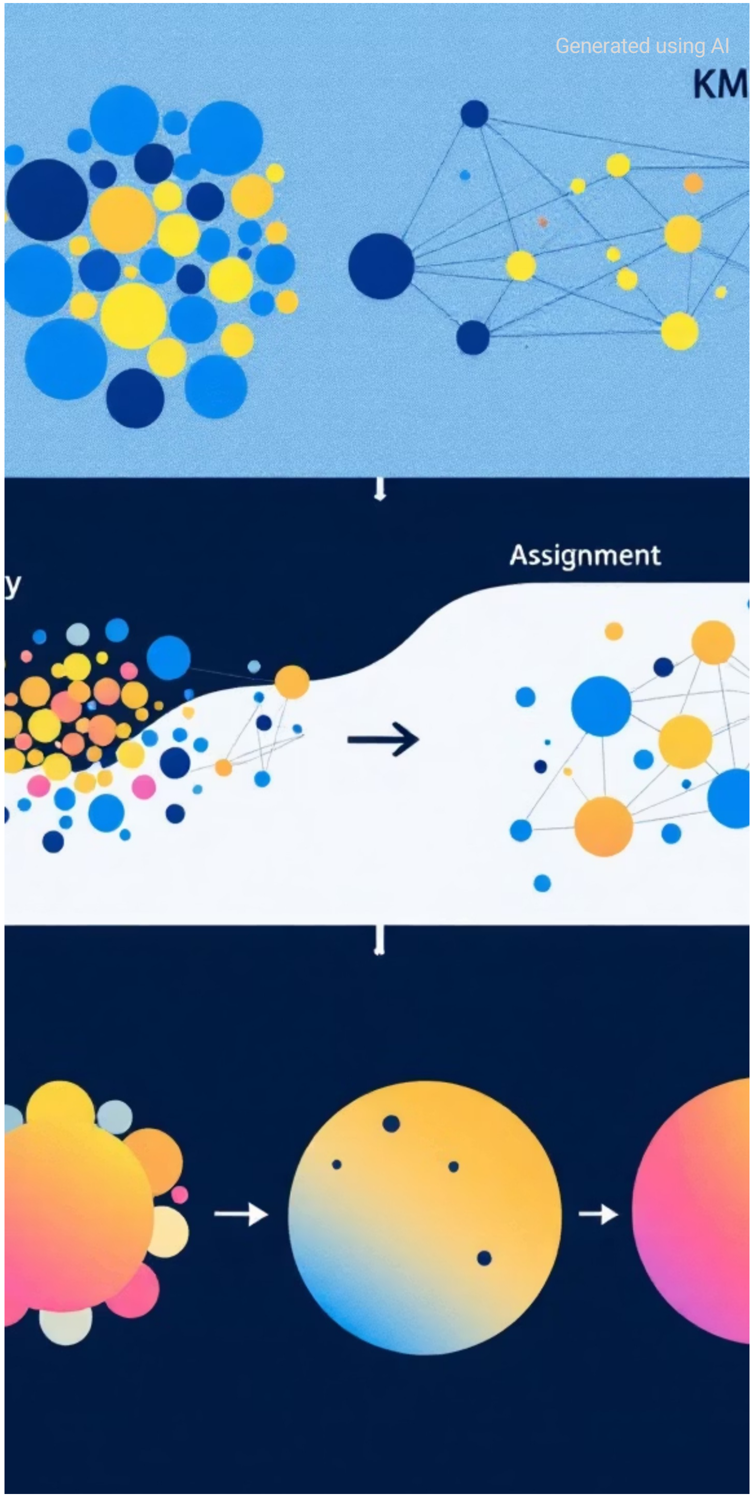
Clustering quality evaluation using

$$\text{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{a_i}{2}+\sum_j\binom{b_j}{2}\right] - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_j}{2}\right]/\binom{n}{2}} \qquad \text{NMI}(U,V) = \frac{2\cdot I(U;V)}{H(U)+H(V)}$$

$s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$ $a(i)$ = mean intra-cluster distance, $b(i)$= mean nearest-cluster distance

- Computational performance measured via wall-clock time with millisecond precision.

ARI: L. Hubert et al, 1985; *NMI:* Scikit-Learn, *Silhouette Score:* Peter. J.R., 1987



MLX Implementation Challenges

Challenge:

No Boolean Indexing

Problem: MLX lacks direct support for operations like X[labels==k].

Solution: Implemented mask-based weighted operations to achieve equivalent functionality.

Challenge:

Limited Random API

Problem: No equivalent to NumPy's np.random.choice().

Solution: Used mx.random.permutation() followed by slicing for centroid initialization.

Challenge:

Type System Differences

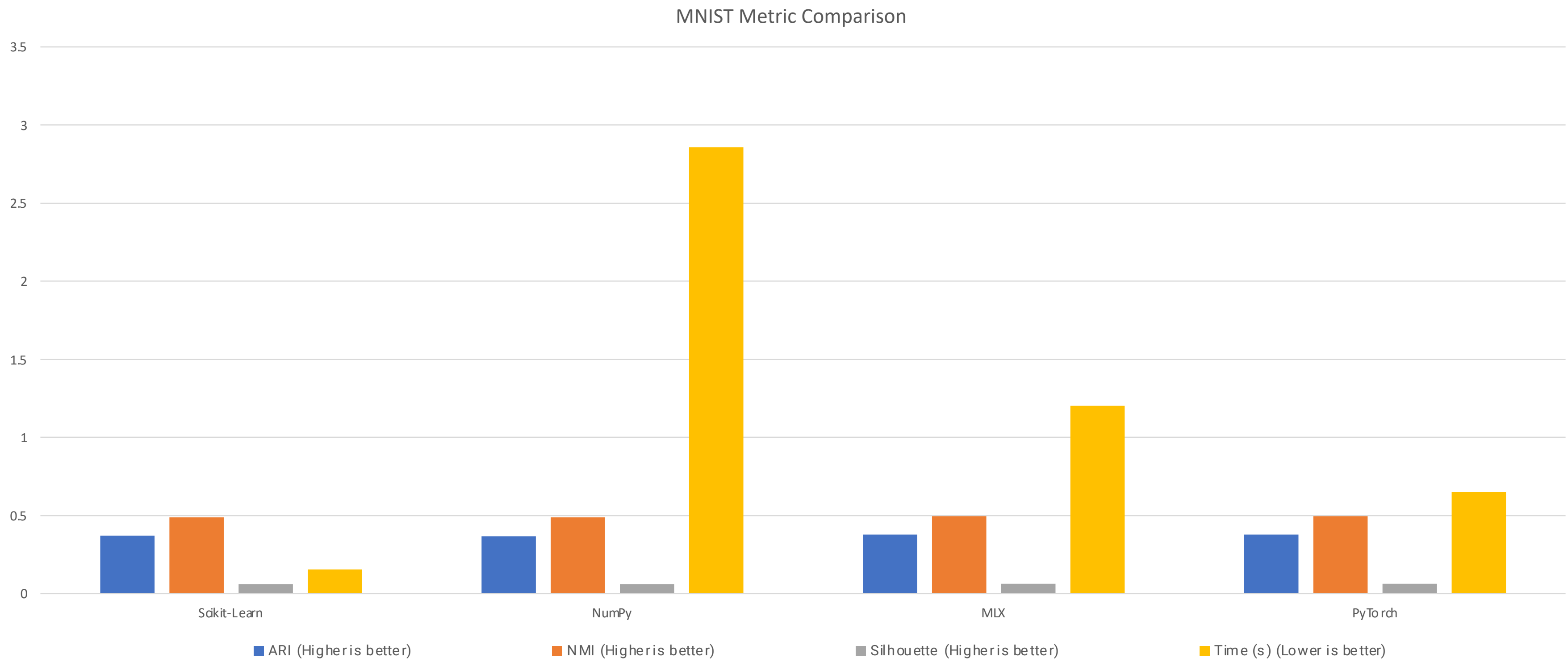
Problem: MLX arrays require careful type checking and explicit conversions.

Solution: Implemented comprehensive input validation and type conversion logic throughout.

These API differences reveal MLX's current difference in their array operations due to lazy computation.

Preliminary Results: MNIST Dataset

Configuration: 7,500 samples with 784 features, k=10 clusters, 20 experimental rounds with seeds 44-63



Clustering Quality

All frameworks achieve equivalent quality metrics within 5%. MLX achieves the **highest ARI (0.3777) and Silhouette (0.0636)**, demonstrating superior clustering results.

ARI (Adjusted Rand Index) *measures similarity between true and predicted clusters, ranging from -1 to 1, where 1 indicates perfect matching.*

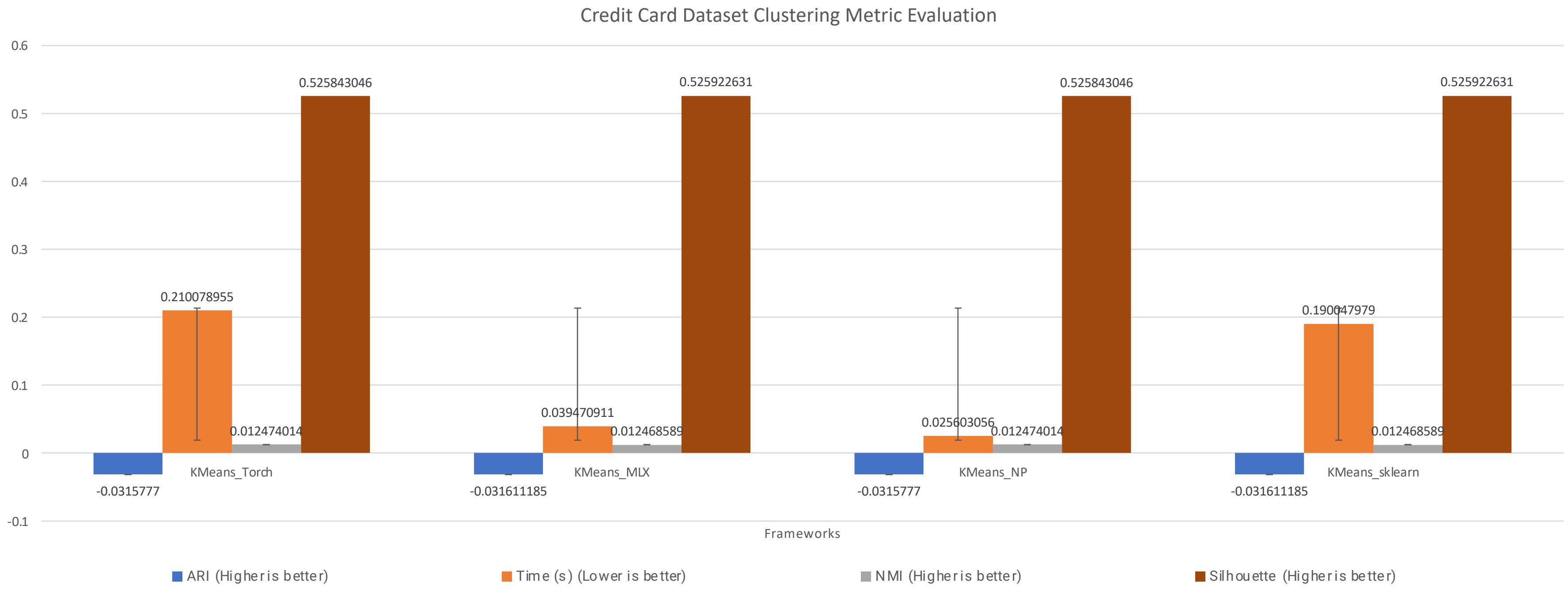
Silhouette Score *measures how similar an object is to its own cluster compared to other clusters, ranging from -1 to 1, with higher values indicating better-defined clusters.*

Performance Ranking

scikit-learn (0.16s) > PyTorch (0.61s) > MLX (1.10s) > NumPy (2.78s). MLX achieves **60% speedup over NumPy**.

Final Results: Credit Card Dataset

Configuration: 30,000 samples with 9 features, k=2 clusters, 500 KMeans rounds with seed 44



Clustering Quality

- **H1 Refuted - Clustering Does Not Align with Churn**
- **Silhouette Score:** ~0.52 for all implementations
- **No correlation** with the hidden churn labels (avg. **ARI** ≈ -0.03).

- **H2 Supported – MLX is Highly Memory Efficient**
- MLX used less memory than sklearn and numpy.

- **Performance Ranking is Data-Dependent**

Performance Ranking

NumPy (0.0245s) > MLX (0.0401s) > sklearn (0.1805s) > PyTorch (0.1955s)

Implications & Recommendations

For Research Community

- MLX is a viable, lightweight, and memory-efficient framework for classical ML algorithms, not just LLMs.

For Practitioners

- K-Means should **not** be used as a proxy for churn prediction on ***this*** dataset. The discovered segments, while statistically valid, are not relevant to ***this*** specific business problem.
- For small-to-medium datasets where **memory efficiency is critical** (e.g., on-device or shared-resource environments), **MLX is a superior choice** over sklearn.

For Future Work

- Investigate why the clusters failed to capture churn (e.g., feature engineering, feature selection)
- Benchmark scaling.

This research fills a critical gap in machine learning systems literature by providing empirical evidence that MLX can effectively support traditional algorithms, not just deep learning, on Apple Silicon hardware.

Thank You