Understanding Why Lloyd's Algorithm in K-Means Clustering Frequently Stalls at Suboptimal Cluster Solutions: Case study in Customer segmentation.

Rifat Rayhan ROMY
Faculty Of Business Administration in Foreign Languages, Bucharest University of Economic Studies, Romania
romyrifat21@stud.ase.ro

Abstract. *Customer segmentation is one of the most consequential analytical tasks in retail management, and Lloyd's K-Means algorithm remains its dominant computational implementation. HoIver, Lloyd's algorithm carries an inherent and under-quantified vulnerability: because the within-cluster sum of squares objective is non-convex, the algorithm is guaranteed to converge only to a local minimum, and the specific solution it returns depends entirely on the random placement of its initial centroids. This paper investigates the severity, structure, and practical consequences of this stalling phenomenon using real point-of-sale transaction data from a single branch of a major European retail chain operating in Southern Poland. Drawing on 162,926 cleaned transactions across three observation windows, I engineered ten transaction-level features and employed a paired Real Data Block and Generated Data Block experimental design that isolates algorithmic stalling from genuine data ambiguity. Using a custom implementation of Lloyd's algorithm that records Mean Centroid Displacement at every iteration, I run 1,000 independent Monte Carlo replications per condition across eleven values of k (2 to 12) and two initialization strategies, totaling 450,000 algorithm executions. I introduce a novel bias metric, Lloyd Bias, defined as mean MCD(k) divided by mean MCD(k_true), which anchors all comparisons to Lloyd's own empirical baseline at the correct number of clusters. Results show that stalling is universal at 99.9% of all runs across all k values when measured within conditions, that cross-run solution consistency collapses from ARI 0.98 at k=2 to ARI 0.49 at k=7, and that K-Means++ initialization provides negligible stall reduction despite its theoretical guarantees. These findings challenge the routine practice of relying on single-run Lloyd segmentation in commercial analytics and establish MCD-based bias as a more principled diagnostic than raw within-cluster sum of squares comparisons.*

Keywords: Lloyd's algorithm, K-Means clustering, local optima, customer segmentation, Monte Carlo simulation, mean centroid displacement, stalling, initialization sensitivity.

Introduction

Customer segmentation, the partitioning of a retailer's transaction base into groups of homogeneous behavior is among the most practically consequential applications of unsupervised machine learning in commerce. The algorithm that underpins this practice in the overwhelming majority of commercial deployments is Lloyd's K-Means (Lloyd, 1982; MacQueen, 1967): a procedure of disarming simplicity that assigns data points to their nearest centroid, recomputes centroids as cluster means, and iterates until convergence. Its dominance is Ill-documented; as Xu and Tian (2015) observe, K-Means remains the default clustering method in virtually all commercial analytics platforms despite two decades of research into its limitations.

Chief among those limitations is the algorithm's susceptibility to suboptimal local minima. Because the sum of squared distances objective is non-convex, Lloyd's algorithm cannot guarantee that it finds the globally best partition of the data — it finds the best partition reachable from wherever its centroids happened to start. Two runs of the identical algorithm on the identical data with different random seeds may, and frequently do, converge with qualitatively different

segmentations. For a practitioner making pricing, assortment, or campaign targeting decisions on the basis of a single segmentation run, this variability is not a theoretical curiosity but a source of genuine decision risk.

The stalling problem is theoretically Ill understood (Selim and Ismail, 1984; Dasgupta, 2008), and the K-Means++ initialization heuristic of Arthur and Vassilvitskii (2007) was specifically designed to reduce its severity by spreading initial centroids more intelligently. Yet the practical magnitude of stalling on real retail transaction data — measured not through abstract worst-case bounds but through empirical distributions across thousands of independent runs — has not been systematically quantified. This paper fills that gap.

I studied stalling using 162,926 real point-of-sale transactions from a single supermarket branch operated by a major European retail chain in Southern Poland, published by Antczak and Iron (2019). Ten features are engineered capturing operational tempo, basket composition, temporal context, and payment behavior. The central research question is: *how severe, how consistent, and how structurally organized is Lloyd's algorithm stalling across the segmentation granularities (values of k) commonly used in retail practice, and does K-Means++ initialization substantively resolve it?*

To answer this question, I introduce three contributions. First, a *paired Real Data Block and Generated Data Block (RDB/GDB) experimental design* that isolates algorithmic stalling from genuine data ambiguity by pairing real statistical fingerprints with synthetic data of known cluster structure. Second, a custom Lloyd's algorithm implementation that records the full *Mean Centroid Displacement (MCD)* trajectory at every iteration, enabling direct observation of convergence dynamics rather than relying on the With-Cluster Sum of Squares, which is incomparable across different values of k. Third, a novel *Lloyd Bias metric* defined as mean_MCD(k) / mean_MCD(k_true) that anchors all comparisons to Lloyd's own empirical baseline, making bias assessments across the full range of k values principled and interpretable.

The paper is structured as follows. Section 2 reviews the relevant literature on K-Means convergence, initialization sensitivity, and retail customer segmentation. Section 3 describes the dataset and experimental methodology in detail. Section 4 presents and discusses the results of the Monte Carlo experiment. Section 5 draws conclusions and outlines directions for further research.

Literature review

Lloyd's algorithm and its convergence properties

Lloyd's algorithm was formulated by Stuart P. Lloyd at Bell Laboratories in 1957 as a procedure for optimal pulse-code modulation design but remained as an internal technical report until its publication in 1982 (Lloyd, 1982). Independently, MacQueen (1967) described a statistically equivalent iterative procedure under the name K-Means. The two contributions are now treated as the same algorithm in the machine learning literature, referred to interchangeably as Lloyd's algorithm or standard K-Means. Its operation consists of three repeating steps: random centroid initialization, point assignment to nearest centroid by Euclidean distance, and centroid update to the cluster mean. Convergence is guaranteed because the within-cluster sum of squares is bound below by zero and decreases at every step — but convergence to the global minimum is not guaranteed, because the objective is non-convex.

The theoretical foundations of this non-convergence Ire established by Selim and Ismail (1984), who proved that Lloyd's algorithm converges to a stationary point of the WCSS objective but that stationary points may be local rather than global minima. The hardness of finding the global

K-Means optimum was subsequently shown to be NP-hard in general by Dasgupta (2008), providing a complexity-theoretic explanation for why no polynomial-time algorithm can guarantee the global solution. Vattani (2011) proved that Lloyd's algorithm can require an exponential number of iterations in the worst case, though such pathological cases rarely arise in practice.

Initialization strategies and K-Means++

The dominant practical response to Lloyd's initialization sensitivity has been the development of better seeding strategies. Arthur and Vassilvitskii (2007) introduced K-Means++, which selects each successive initial centroid with probability proportional to its squared distance from already-chosen centroids, thereby spreading seeds across the data space. Their theoretical guarantee — that K-Means++ achieves expected WCSS at most O(log k) times the global optimum, compared to unbounded expected performance under random initialization — made it the de facto standard in most implementations.

HoIver, the practical performance of K-Means++ relative to random initialization varies considerably across datasets. Celebi et al. (2013) conducted a systematic comparison across fourteen real-world datasets and found that K-Means++ consistently reduces WCSS variance but does not eliminate it, and that for some datasets the improvement over random initialization is negligible. Nie et al. (2022) proposed a reformulation of the K-Means objective to improve algorithmic efficiency, while Zhang et al. (2025) specifically addressed residual local-optimality gaps in Lloyd's convergence proofs. Brunsch and Röglin (2013) shoId that K-Means++ initialization can itself lead to suboptimal solutions in adversarially constructed cases. Grunau and Rozhoň (2022) further demonstrated that Lloyd's algorithm remains susceptible to local optima even under improved initialization when data contains outliers — a condition frequently present in real transaction data.

Customer segmentation in retail

The application of clustering to retail customer segmentation has been studied extensively, with RFM modelling (Recency, Frequency, Monetary value) establishing the dominant feature framework. Tabianan et al. (2022) applied K-Means to purchase behavior data and demonstrated the practical relevance of segment quality for personalized retail strategies. Ogunleye et al. (2023) compared multiple clustering algorithms on UK retail RFM data and found that K-Means consistently produces interpretable solutions but exhibits sensitivity to initialization at higher segment granularities. Joung and Kim (2023) studied interpretability in machine-learning-based segmentation, noting that the same algorithm applied to the same data could yield different customer profiles across runs.

Recent work has further connected segmentation quality to downstream business outcomes. Rungruang et al. (2024) demonstrated that hierarchical segmentation quality degrades when underlying clustering is unstable. Kasem et al. (2024) shoId that sales prediction accuracy in direct marketing depends critically on segmentation consistency over time. Liu et al. (2025) applied Q-learning augmented K-Means to digital marketing segmentation, specifically to address the instability problem, while Jiang et al. (2025) embedded K-Means within a hybrid RFM and deep learning pipeline for churn prediction. These studies collectively demonstrate that Lloyd's stalling has practical downstream consequences, yet none quantifies its severity using an MCD-based framework on real POS transaction data.

Research gap and contribution

Three specific gaps motivate the present study. First, existing empirical stalling analyses use WCSS as their primary metric, which is inherently incomparable across different values of k because

WCSS decreases monotonically as k increases regardless of solution quality. Second, the RDB/GDB design — pairing real data statistical properties with synthetic ground-truth structure — has not been applied to isolate algorithmic stalling from genuine cluster ambiguity in a retail context. Third, while cross-run solution consistency (measured by Adjusted Rand Index between independent runs) is arguably more relevant to business practice than solution quality for a single run, it has received limited empirical attention. This paper addresses all three gaps using 162,926 real Polish supermarket transactions, 1,000 Monte Carlo replications per condition, and a novel MCD-based bias metric.

Methodology

Dataset and feature engineering

The empirical foundation of this study is the Polish Supermarket POS Dataset published by Antczak and Iron (2019) under a Creative Commons CC-BY license (MDPI *Data*, 4(2), 67; DOI: 10.3390/data4020067). The dataset contains unmodified point-of-sale transaction logs from a single branch of a major European retail chain in Southern Poland, spanning three non-contiguous two-Iek observation windows: December 7–19, 2017 (66,863 transactions); February 13–26, 2019 (47,066 transactions); and March 28 – April 10, 2019 (49,340 transactions). After removing void transactions, cancelled sales, and system entries by applying the criteria Amount > 0, ArtNum > 0, and TranTime > 0, the cleaned dataset contains 162,926 transactions with zero missing values across all retained fields.

Ten analytical features are engineered from the raw POS log fields, spanning four behavioral dimensions. Operational tempo is captured by TranTime (total transaction duration in seconds from first scan to payment) and BreakTime (idle time betIen consecutive item scans). Basket composition is represented by ArtNum (number of items), Amount (transaction value in Polish Złoty), and amount_per_item (mean item value = Amount / ArtNum). Temporal context is encoded by hour (hour of day, range 6–22), day_of_Iek (0=Monday through 6=Sunday), and is_Iekend (binary indicator). Payment behavior is represented by payment_card and payment_cash (binary indicators). This feature set deliberately combines continuous, count, and binary distributions — precisely the heterogeneous geometry that maximizes the risk of Lloyd's random seeds landing in uninformative regions of the feature space.

Table 1 presents descriptive statistics for the ten features across the full 162,926-transaction dataset. The high coefficient of variation observed for BreakTime (CV = 1.72), amount_per_item (CV = 3.64), and is_Iekend (CV = 1.70) reflects the skeId and heterogeneous distributions that characterise real retail transaction data and create challenging conditions for Lloyd's centroid initialization.

Table 1. Descriptive statistics of the ten engineered POS features (n = 162,926 transactions)

Feature	Min	Mean	Median	Max	Std	CV
TranTime (sec)	1	71.4	55	1,137	60.6	0.85
BreakTime (sec)	-12	41.9	21	1,199	72.2	1.72
ArtNum	1	15.5	10	422	16.3	1.05
Amount (PLN)	0.08	72.6	45.9	6,884	87.0	1.20
amount_per_item	0.03	5.83	4.13	6,884	21.2	3.64
hour	6	13.9	14	22	3.85	0.28
day_of_Iek	0	2.95	3	6	1.88	0.64
is_Iekend	0	0.26	0	1	0.44	1.70
payment_card	0	0.48	0	1	0.50	1.04
payment_cash	0	0.52	1	1	0.50	0.96

Source: Authors' own research results.

The RDB/GDB experimental design

A fundamental challenge in studying clustering algorithm behavior on real data is the absence of ground truth: I cannot know the true number of clusters in the 162,926 transactions, and without ground truth I cannot compute cluster recovery accuracy (Adjusted Rand Index, ARI) or define what 'correct-k' means in an operationally precise way. The Real Data Block and Generated Data Block (RDB/GDB) design solves this problem by separating the statistical properties of the real data from the cluster structure.

Ten independent Real Data Blocks (RDB_01 through RDB_10) are drawn as random samples of 300 transactions from the cleaned POS dataset. For each RDB, per-feature statistics are computed: minimum, maximum, mean, standard deviation, and coefficient of variation (CV = σ/μ). These statistics constitute a statistical fingerprint of the real data.

For each RDB, a corresponding Generated Data Block (GDB) is synthesized to: (i) embed exactly k_true = 7 Gaussian clusters with known ground-truth labels; (ii) match the paired RDB's per-feature [min, max] range exactly through value clipping; and (iii) approximate the RDB's per-feature CV through calibration of cluster standard deviations. Seven cluster centers are placed uniformly across each feature's range, rows are shuffled, and labels are withheld from Lloyd's algorithm. The ground-truth labels are used only for post-hoc ARI computation. The design principle is that any instability Lloyd's algorithm exhibits on a GDB cannot be attributed to genuine data ambiguity, because the true cluster structure is known and Ill-separated: all observed stalling is algorithmic in origin.

Lloyd's algorithm and the MCD bias metric

A custom implementation of Lloyd's algorithm, designated run_lloyd(), was developed in Python to record the full centroid displacement trajectory at every iteration. At each iteration t, Mean Centroid Displacement is computed as:

$$MCD(t) = (1/k) \times \Sigma_i \, \| \, c_i(t) - c_i(t-1) \, \|_2$$

where $c_i(t)$ denotes the position of centroid i at iteration t and k is the number of clusters. This measure is preferred over WCSS because WCSS decreases monotonically as k increases, making values incomparable across different k. MCD measures the geometric convergence effort per centroid per iteration and is naturally comparable across k values.

The primary analytical metric, *Lloyd Bias*, is defined as:

$$Lloyd\_bias(k) = mean\_MCD\_Lloyd(k) / mean\_MCD\_Lloyd(k\_true)$$

The denominator is pre-computed by running Lloyd's algorithm 1,000 times at k_true = 7 on each GDB block and averaging the resulting MCD values. A bias value greater than 1.0 indicates that Lloyd at k demands proportionally more centroid movement than at the correct-k baseline, reflecting under-segmentation or poor initialization. A bias below 1.0 indicates over-segmentation — artificial fragmentation into many small, tight clusters where centroids barely need to move. K-Means++ bias values are also divided by the same Lloyd reference, ensuring a fair comparison: K-Means++ is evaluated against Lloyd's own correct-k standard.

Monte Carlo experiment design

The full experiment covers 220 conditions: 10 GDB blocks × 11 k values (2 to 12) × 2 algorithms (Lloyd and K-Means++). Each condition is replicated 1,000 times with random seeds 0 to 999, using n_init = 1 to force single-initialization and expose stalling directly. This yields 220,000 Lloyd runs and 220,000 K-Means++ runs, plus 10,000 reference pre-computation runs at k_true = 7, for a total of 450,000 algorithm executions. The experiment was implemented in Python 3.12 using NumPy 2.0.2, Pandas 2.2.2, and scikit-learn 1.6.1. Total computation time was approximately 7,030 seconds on a single CPU core.

For each condition, nine metrics are recorded: lloyd_bias (primary bias metric), mcd_cv (coefficient of variation of mcd_mean across 1,000 runs, the stalling fingerprint), stall_rate (fraction of runs with mcd_mean > 1.05 × minimum mcd_mean in condition), lloyd_stall_rate (fraction of runs that cannot match Lloyd's correct-k MCD), ari_cross_mean (mean ARI between 100 random pairs of independent runs, measuring solution consistency), ari_truth_mean (mean ARI betIen Lloyd's labels and GDB ground-truth labels), mcd_first_mean (MCD at iteration 1, measuring initialisation quality), n_iter_mean (mean iterations to convergence), and supplementary cluster quality metrics (Silhouette Score, Davies-Bouldin Index, Calinski-Harabász Index).

Results and discussions

Lloyd bias across k values

Table 2 presents the full Lloyd Bias results averaged across the 10 GDB blocks, together with stall rates and cross-run ARI for both Lloyd's algorithm and the K-Means++ benchmark. The bias pattern shows a clear three-region structure: a monotonically decreasing above-1.0 region for k < 7 (under-segmentation), the reference value at k = 7, and a below-1.0 region for k > 7 (over-segmentation). In the under-k region, Lloyd bias reaches 1.9180 at k = 2, reflecting that with only two clusters each centroid must span an extremely heterogeneous region of the 10-dimensional transaction feature space, requiring disproportionate displacement effort. As k approaches k_true = 7, bias approaches 1.0 monotonically, with k = 6 yielding 1.0197. In the over-k region, bias falls to 0.9750 at k = 11, reflecting artificial deflation as clusters become too small to require substantial centroid movement.

Table 2. Lloyd Bias, stall rate, and cross-run ARI by k (averaged across 10 GDB blocks, 1,000 replications per condition)

k	Region	Lloyd Bias	KM++ Bias	Stall %	Lloyd Stall %	MCD CV	ARI-cross (Lloyd)	ARI-cross (KM++)
2	Under-k	1.9180	1.8590	99.7	99.6	0.2138	0.9827	0.9903
3	Under-k	1.2622	1.2393	99.9	72.0	0.3111	0.5540	0.5603
4	Under-k	1.0959	1.1012	99.9	57.3	0.2823	0.5218	0.5366
5	Under-k	1.0492	1.0621	99.9	52.6	0.2641	0.5096	0.5359
6	Under-k	1.0197	1.0482	99.8	48.9	0.2502	0.4954	0.5332
7 ★	Correct-k	1.0000	1.0358	99.9	46.5	0.2401	0.4898	0.5344
8	Over-k	0.9943	1.0357	99.8	46.0	0.2348	0.4832	0.5277
9	Over-k	0.9916	1.0407	99.8	45.5	0.2337	0.4823	0.5253
10	Over-k	0.9802	1.0373	99.8	43.8	0.2352	0.4830	0.5235
11	Over-k	0.9750	1.0379	99.8	42.7	0.2275	0.4773	0.5305
12	Over-k	0.9781	1.0392	99.8	44.5	0.2250	0.4837	0.5289

Source: Authors' own research results. ★ = k_true reference point. Stall % = within-condition; Lloyd Stall % = vs Lloyd correct-k baseline.

The universal stalling finding

The most striking and unexpected finding in Table 2 is the within-condition stall rate column: Lloyd's algorithm stalls in 99.7–99.9% of all runs across all values of k. This result requires careful interpretation. The within-condition stall rate is defined as the fraction of runs whose mcd_mean exceeds 1.05 times the minimum mcd_mean observed in the same (block, k) condition. The near-universality of stalling therefore means that in almost every condition, the best solution found across 1,000 runs is substantially better than the typical solution — i.e., most runs converge to worse local optima than the best achievable. This is not a computational pathology but a direct consequence of the highly heterogeneous 10-dimensional feature space: with features spanning different scales and distributions, random centroid placement almost always places at least one centroid in an uninformative region, and the algorithm converges from there rather than finding the globally better solution.

The Lloyd-referenced stall rate — defined as the fraction of runs whose mcd_mean exceeds Lloyd's own mean MCD at k_true = 7 — shows a more structured pattern. At k = 2, 99.6% of runs fail to reach Lloyd's correct-k performance standard, which is unsurprising because two-cluster solutions require far more per-centroid displacement in a seven-cluster world. The Lloyd-referenced stall rate decreases from 99.6% at k = 2 to 46.5% at k = 7 (the reference point), reflecting that as k approaches the true number of clusters, Lloyd is increasingly likely to achieve convergence comparable to its correct-k behavior.

Solution consistency collapse

The cross-run ARI column reveals the most practically important finding. At k = 2, the mean ARI between pairs of independent Lloyd runs is 0.9827 — near-perfect consistency, indicating that almost every random initialization leads to the same two-cluster partition. This is expected: the solution space for two clusters is nearly trivial. HoIver, ARI collapses to 0.4898 at k = 7 and remains near 0.48–0.49 across the entire over-k region (k = 8 to 12). An ARI of approximately 0.49

means that two independent runs of Lloyd's algorithm on the same data with the same k produce a partition that agrees with the other only as much as would be expected from a moderately informed but inconsistent process — far from the deterministic, reproducible outputs that retail segmentation decisions require.

The sharpest ARI drop occurs between k = 2 and k = 3, from 0.9827 to 0.5540 — a collapse of 0.43 ARI points over a single increment in k. This indicates that as soon as the solution space becomes non-trivial (three or more clusters), Lloyd's random initialization produces qualitatively different results depending on the seed. The total ARI collapse from k = 2 to k = 12 is 0.4990 points. For a practitioner who runs their segmentation algorithm once and bases targeting, pricing, or assortment decisions on the result, this means there is approximately a 50% chance that a different run would produce a substantively different set of customer segments — with no indication from the algorithm itself of which run is correct.

K-Means++ benchmark comparison

The K-Means++ results expose a paradox. The K-Means++ bias at k = 7 is 1.0358, meaning that K-Means++ actually achieves slightly *higher* mean MCD than Lloyd at the correct-k reference. This counter-intuitive result follows from the structure of the GDB: because clusters are constructed with uniform centers placed across the feature range, K-Means++'s distance-proportional seeding spreads centroids further than is optimal for this particular geometry, leading to slightly more centroid movement per iteration than Lloyd's random seeds. More importantly, K-Means++ within-condition stall rates are essentially identical to Lloyd's (99.8% versus 99.9% at k = 7), and K-Means++ cross-run ARI (0.5344 at k = 7) is only marginally higher than Lloyd's (0.4898). The stall reduction from K-Means++ is 0.1 percentage points — negligible.

This finding challenges the widespread assumption that K-Means++ resolves the stalling problem. While K-Means++ does improve MCD CV slightly (0.2351 versus 0.2401 at k = 7), it does not prevent Lloyd's algorithm from converging to a wide variety of different local optima. The theoretical O(log k) guarantee provides an upper bound on expected WCSS but does not prevent the solution consistency collapse that is the most practically damaging aspect of Lloyd's stalling. Grunau and Rozhoň (2022) reached similar conclusions: K-Means++ reduces worst-case expected cost but does not resolve the inconsistency problem on realistic heterogeneous data.

MCD variability and convergence dynamics

Table 3 reports the MCD coefficient of variation at each k for both algorithms, together with the mean MCD at iteration 1 (mcd_first, a direct measure of initialization quality) and mean iterations to convergence. The MCD CV is highest at k = 3 (0.3111 for Lloyd, 0.3034 for K-Means++) and decreases steadily toward k = 12 (0.2250 and 0.2185 respectively). This monotonic decrease in MCD CV as k increases, despite increasing within-condition stall rates, reflects that over-segmented solutions are geometrically more similar to each other — many small clusters all look roughly the same — even though they are partitioning data differently as measured by ARI.

Table 3. MCD variability, initialization quality, and convergence cost by k (Lloyd's algorithm, averaged across 10 GDB blocks)

k	MCD CV (Lloyd)	MCD CV (KM++)	CV Diff	mcd_first (Lloyd)	mcd_first (KM++)	n_iter (Lloyd)
2	0.2138	0.2015	+0.0123	higher	lower	more
3	0.3111	0.3034	+0.0078	—	—	—
7 ★	0.2401	0.2351	+0.0050	—	—	—
10	0.2352	0.2221	+0.0131	—	—	—
12	0.2250	0.2185	+0.0066	lower	lowest	fewest

Source: Authors' own research results. ★ = k_true reference. Direction of mcd_first and n_iter relative to k=7 reference.

Statistical match verification

Table 4 presents the statistical match between RDB_01 and GDB_01, verifying that the GDB faithfully reproduces the real data's statistical fingerprint. Feature ranges match exactly by construction (clipping enforces this). CV differences are small for operationally stable features: TranTime (0.0539), hour (0.0909), and day_of_week (0.1589). Larger CV differences appear for binary and highly skewed features: is Weekend (0.9917), BreakTime (0.6878), and amount_per_item (0.5458). These differences are expected and unavoidable — Gaussian mixtures cannot perfectly reproduce binary or heavily right-skewed empirical distributions — but they confirm that the GDB is matched to the real data's difficulty level, not simplified relative to it.

Table 4. Statistical match verification: RDB_01 versus GDB_01 (n = 300 each)

Feature	RDB_01 range	GDB_01 range	RDB CV	GDB CV	|ΔCV|
TranTime	[4.0, 519.0]	[4.0, 519.0]	0.8828	0.8289	0.054
BreakTime	[2.0, 569.0]	[2.0, 569.0]	1.6055	0.9177	0.688
ArtNum	[1.0, 100.0]	[1.0, 100.0]	1.0352	0.8939	0.141
Amount	[0.1, 545.9]	[0.1, 545.9]	1.1545	0.9149	0.240
amount_per_item	[0.1, 97.3]	[0.1, 97.3]	1.6208	1.0750	0.546
is_Iekend	[0.0, 1.0]	[0.0, 1.0]	1.9625	0.9708	0.992

Source: Authors' own research results.

Conclusion

This paper investigated the severity, structure, and practical consequences of Lloyd's K-Means algorithm stalling at suboptimal local minima, using 162,926 real point-of-sale transactions from a Polish supermarket chain and 450,000 algorithm executions across a 1,000-replication Monte Carlo experiment. Four principal findings emerge.

First, stalling is effectively universal in the within-condition sense: across all k values and both initialization strategies, 99.7–99.9% of individual Lloyd runs converge to a solution that is measurably worse than the best solution findable from a different random seed. This result establishes that the default practice of running Lloyd's algorithm once — or even a small number of times — is statistically indefensible for any segmentation task where solution quality matters.

Second, the Lloyd Bias metric reveals a clear three-region structure that is grounded in Lloyd's own convergence dynamics rather than an abstract structural ratio. Under-segmentation ($k < k\_true$) inflates bias to 1.92 at $k = 2$, reflecting the disproportionate convergence effort required when too few centroids must cover heterogeneous data regions. Over-segmentation ($k > k\_true$) deflates bias below 1.0 as small dense clusters require minimal centroid movement — but stall rates do not fall correspondingly, confirming that the over-k region is problematic in a different way: many different suboptimal solutions exist and Lloyd finds a different one almost every time.

Third, solution consistency, measured as cross-run ARI, collapses from 0.98 at $k = 2$ to 0.49 at $k = 7$ — the value most commonly used in retail segmentation. This means that two independent runs of Lloyd's algorithm on the same customer data produce partitions that agree with each other only about as Ill as a moderately informed random assignment. The consistency collapse is rapid: the largest single drop occurs betIen $k = 2$ and $k = 3$, implying that even three-cluster segmentations are affected.

Fourth, K-Means++ initialization provides negligible practical improvement over random initialization. Within-condition stall rates are 99.8% for K-Means++ versus 99.9% for Lloyd at $k = 7$ — a reduction of 0.1 percentage points. Cross-run ARI at $k = 7$ is 0.534 for K-Means++ versus 0.490 for Lloyd — an improvement of 0.044 ARI points that falls far short of the consistency levels required for reliable commercial segmentation. The counter-intuitive finding that K-Means++ achieves slightly higher Lloyd Bias than random initialization at $k = 7$ reflects a specific feature of the GDB construction geometry, but the broader pattern is clear: smarter seeding does not resolve the solution inconsistency problem.

The primary contribution of this paper is the introduction of the Lloyd Bias metric and the RDB/GDB design, which together provide a principled and interpretable framework for quantifying algorithmic stalling on realistically structured data. The finding that MCD-based bias provides a coherent and comparable signal across all k values, while WCSS does not, has implications beyond this study: it suggests that MCD tracking should become a standard diagnostic in any clustering pipeline where solution reliability matters. Several limitations should be acknowledged: the GDB construction uses Gaussian cluster models, which may not capture all forms of cluster structure present in real transaction data; the study is based on a single retail chain and country; and the computational intensity of 1,000 replications per condition makes direct replication on very large datasets challenging without HPC resources. Future work should examine whether the stalling patterns observed here — particularly the universality of within-condition stalling and the ARI collapse — are consistent across different retail contexts, feature spaces, and distance metrics.

References

Antczak, T., & Iron, R. (2019). POS transaction data from a supermarket in Southern Poland. Data, 4(2), 67. https://doi.org/10.3390/data4020067

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), 1027–1035.

Brunsch, T., & Röglin, H. (2013). A bad instance for k-means++. Theoretical Computer Science, 505, 19–26. https://doi.org/10.1016/j.tcs.2013.05.031

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Systems with Applications, 40(1), 200–210.

Dasgupta, S. (2008). The hardness of k-means clustering. Technical Report CS2007-0890, UC San Diego.

Grunau, C., & Rozhoň, V. (2022). Adapting k-means algorithms for outliers. Proceedings of Machine Learning Research, 162, 7845–7886.

Jiang, Z., Hu, Y., & Liu, Z. (2025). Enhancing customer retention in online retail through churn prediction: A hybrid RFM, K-means, and deep neural network approach. Expert Systems with Applications. https://doi.org/10.1016/j.eswa.2025.127441

Joung, J., & Kim, H. (2023). Interpretable machine learning-based approach for customer segmentation for new product development from online product reviews. International Journal of Information Management, 70, 102641.

Kasem, M. S., Hamada, M., & Taj-Eddin, I. (2024). Customer profiling, segmentation, and sales prediction using AI in direct marketing. Neural Computing and Applications, 36(9), 4995–5005.

Liu, X., Li, Y., Wang, Z., Feng, J., Li, J., & Hao, T. (2025). Customer segmentation in digital marketing using a Q-learning based differential evolution algorithm integrated with K-means clustering. PLOS ONE, 20(2), e0318519.

Lloyd, S. P. (1982). Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2), 129–137.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, 281–297.

Nie, F., Li, Z., Wang, R., & Li, X. (2022). An effective and efficient algorithm for K-Means clustering with new formulation. IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2022.3155456

Ogunleye, B., John, J. M., & Zuva, T. (2023). An exploration of clustering algorithms for customer segmentation in the UK retail market. Analytics, 2(4), 809–823.

Rungruang, C., Riyapan, P., Intarasit, A., & Muangprathub, J. (2024). RFM model customer segmentation based on hierarchical approach using FCA. Expert Systems with Applications, 237, 121449.

Selim, S. Z., & Ismail, M. A. (1984). K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(1), 81–87.

Tabianan, K., Velu, S., & Ravi, V. (2022). K-Means clustering approach for intelligent customer segmentation using customer purchase behavior data. Sustainability, 14(12), 7243.

Vattani, A. (2011). k-means requires exponentially many iterations even in the plane. Discrete & Computational Geometry, 45(4), 596–616.

Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. Annals of Data Science, 2(2), 165–193.

Zhang, K., Bharara, S., Moubayed, A., Brown, B., & Zhang, Y. (2025). Modified K-means algorithm with local optimality guarantees. arXiv preprint arXiv:2506.06990.