International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 9 Issue 8 August - 2025, Pages: 80-85

SmartSort: An Intelligent Framework for Optimizing Sorting Efficiency Using AI in Real-Time Systems

Khaleel Alnajjar and Samy S. Abu-Naser

Department of Information Technology
Faculty of Engineering and Information Technology
Al-Azhar University – Gaza
Gaza, Palestine

Abstract: *SmartSort is proposed as an adaptive hybrid sorting algorithm that leverages both QuickSort and HeapSort techniques, augmented by an AI-driven decision module to optimize performance under real-time constraints. This framework addresses the challenges of unpredictable input distributions and stringent timing requirements in real-time systems by combining the speed of QuickSort on average and the guaranteed worst-case behavior of HeapSort. SmartSort uses machine learning classifiers (e.g. decision trees, XGBoost or neural networks) to analyze data patterns – such as size, range, and entropy – and adapt its strategy (pivot selection or algorithm choice) on-the-fly[1][2]. We formally analyze SmartSort's time complexity and space usage, demonstrating average-case O(n log n) and worst-case O(n log n) behavior (due to the HeapSort fallback) while maintaining low overhead. The algorithm's best-, average-, and worst-case behaviors are discussed in detail, and we show that SmartSort can satisfy bounded-latency and predictability requirements typical of real-time systems[3][4]. Recent literature on AI-driven algorithm optimization and real-time scheduling is surveyed to support this design[5][6].*

Keywords : Adaptive Sorting Algorithm , Hybrid Sorting , QuickSort , HeapSort , Real-Time Systems , Algorithm Optimization , Predictability , Bounded Latency.

Introduction

Sorting is a fundamental operation in computer systems, and its efficiency is critical in many applications – from database operations to scheduling tasks in operating systems. In real-time computing, guaranteed timing behavior is paramount: algorithms must respect strict worst-case execution time (WCET) constraints to meet deadlines[7]. Traditional sorts like QuickSort are very fast on average but can degrade to $O(n^2)$ time for certain inputs, making them unpredictable for hard real-time uses[8][3]. HeapSort, by contrast, has worst-case $O(n \log n)$ time but is often slower in practice and less cache-friendly[9][3]. These trade-offs motivate a hybrid approach: use QuickSort's speed on typical data, but fall back on HeapSort to cap worst-case cost. This idea is embodied by the introspective sort (introsort), which begins with QuickSort and switches to HeapSort when recursion depth is high, thus ensuring $O(n \log n)$ worst-case time[3]. SmartSort extends this concept by also incorporating AI-driven adaptivity.

The motivation for SmartSort is twofold. First, real-time systems increasingly encounter dynamic and unpredictable data streams, such as sensor readings or network packets, where data characteristics may vary widely over time. An algorithm that can recognize patterns (e.g. nearly-sorted, heavy-tails, uniform distribution) and adapt accordingly can maintain high efficiency. Second, modern AI and machine learning (ML) techniques offer powerful tools to predict which algorithmic strategy will perform best on a given input. Recent work shows that ML-guided approaches can significantly accelerate sorting: for instance, learned or neural-network-based sorts have achieved substantial speedups on large datasets[2][5].

Thus, SmartSort is designed as an intelligent hybrid algorithm. It combines QuickSort and HeapSort, augmented with a machine-learned classifier that examines features of the input array (size n, value range, entropy, or other statistics) to predict the most efficient strategy. For example, if the classifier detects a nearly sorted array (which can trap naive QuickSort), SmartSort might choose a robust pivot or switch early to HeapSort. Conversely, if data appears random, it proceeds with QuickSort to capitalize on its low overhead. This adaptive behavior optimizes sorting efficiency based on actual data patterns, while the hybrid design ensures bounded latency and predictability by avoiding QuickSort's pathological cases[3][4].

In summary, this paper presents the SmartSort framework with a focus on theoretical analysis. We discuss its hybrid design, detail how AI (e.g. classifiers) is integrated for adaptability, analyze its time/space complexity and case performance, and explain how it meets real-time system requirements. We draw on recent research on AI-enhanced algorithms and real-time scheduling to support SmartSort's design choices[1][2].

Problem Statement

Traditional sorting algorithms such as QuickSort and HeapSort are well-established for general-purpose computing; however, their fixed behavior makes them suboptimal in real-time systems where timing constraints and unpredictable input distributions prevail. QuickSort, while efficient on average, suffers from poor worst-case performance, whereas HeapSort provides consistent performance but may be slower in practice.

Real-time systems demand guaranteed predictability, low latency, and adaptability to varying input data. Current sorting techniques fail to dynamically adjust to the input characteristics or execution context, leading to inefficiencies or deadline violations in time-sensitive applications.

Moreover, despite the growing integration of artificial intelligence into various algorithmic domains, AI-driven optimization of core algorithms like sorting remains underexplored, particularly in the context of real-time constraints.

Methodology

The **SmartSort algorithm** operates as follows:

- **Hybrid Quicksort-HeapSort design:** SmartSort begins like a traditional QuickSort using a partition-and-recurse strategy. It selects a pivot (e.g. using median-of-three selection, which is more robust than choosing first or last elements)[14]. After partitioning the array, it recursively sorts each part. However, SmartSort tracks recursion *depth* or partition balance: if the recursion depth exceeds a threshold (typically $O(\log n)$) or if a partition is highly unbalanced, it switches to HeapSort for that (sub)array, ensuring that worst-case time stays $O(n \log n)$[3][4]. This introspective mechanism alone avoids QuickSort's quadratic worst-case.

- **AI-based adaptive strategy:** Crucially, SmartSort includes a machine learning decision module to adapt to data patterns. At runtime, before or during partitioning, it computes features of the subarray (for instance: length n, the range of values, entropy, or indicators of sortedness). These features are fed into a pre-trained classifier (e.g. a decision tree, XGBoost model, or a small neural network). The classifier predicts the best strategy: for example, it may suggest proceeding with QuickSort using a particular pivot rule, or immediately switching to HeapSort. This approach is inspired by Adaptive Hybrid Sort's use of an XGBoost model: in that work, the model (trained on synthetic datasets) achieved 92.4% accuracy in selecting the optimal sort strategy based on features {size, range, entropy}[1]. SmartSort's model can be trained similarly on representative data distributions.

  - *Example:* If the classifier detects that the subarray has very low entropy (e.g. nearly sorted or many repeated values), it might predict that QuickSort performance would be poor (even with median-of-three). SmartSort would then choose HeapSort for this segment. Conversely, if data appears random, the classifier would favor QuickSort for its lower overhead. Table-like decision logic or a forest model can encode such decisions (for instance, "if $n<20$ use insertion sort, else if range > threshold use counting sort" is one rule-based approach). In practice, SmartSort can combine simple static rules for trivial cases (small n, etc.) with ML for larger n[15].
- **Algorithm steps:** The high-level SmartSort procedure can be outlined as:

- **Base cases:** If n is small (e.g. < 16), use Insertion Sort (fast for tiny arrays).
- **Feature extraction:** Compute key statistics of the array (size n, value range, entropy, etc.).
- **Decision:** Use the ML model (or rule) to select strategy:
  - If strategy = QuickSort: pick pivot (e.g. median-of-three) and partition.
  - If strategy = HeapSort: ignore pivot, build a heap and perform HeapSort on this subarray.
- **Recursion:** Apply SmartSort recursively on partitions (if QuickSort) or done (if HeapSort).
- **Fallback:** If using QuickSort, track recursion depth. If depth > $2 \cdot \lfloor \log_2(n) \rfloor$, forcibly switch to HeapSort on the current segment (as in introsort[3]).

This hybrid+AI approach ensures SmartSort **adapts to data**. For instance, genetic algorithms or neural networks might be used to fine-tune pivot choice or sort direction; prior work found that genetic algorithms could optimize parameters for real-time sorting to avoid worst-case patterns[16][6]. By leveraging AI, SmartSort can refine the simple introsort heuristic with data-driven insights. Recent experiments in ML-driven sorting showed large performance gains: e.g., a neural network–based sort was 40% faster than QuickSort on datasets of size $>10^7$[2]. SmartSort aims for similar gains by "learning" from data distributions.

- **Real-time integration:** In a real-time system, SmartSort would typically be part of a task (e.g. reorganizing data for scheduling or analytics). Its hybrid design and AI module must not violate timing constraints. Therefore:
- The ML inference is designed to be extremely fast: models can be quantized or simplified. For example, [15] reports an XGBoost model with 0.2ms per decision and 1MB size, adding only ~0.1% overhead on large sorts[17]. SmartSort would use lightweight models (or simple decision trees) so that classification adds negligible latency. Indeed, for very small subarrays ($n \leq 100$) SmartSort can skip the ML step entirely and rely on fixed rules to minimize overhead[17].
- Space usage is also controlled: the sort is in-place (aside from recursion stack) and the ML model has constant size. Thus, SmartSort does not introduce unbounded memory demands beyond the input and model size.
- These properties (bounded decision time, fixed extra memory) allow SmartSort to maintain predictability. Combined with the worst-case time guarantee (heap fallback), SmartSort can be certified for real-time use like any $O(n \log n)$ algorithm.

In summary, SmartSort's methodology fuses classical algorithm engineering with modern AI. By outlining clear algorithmic steps and ML integration, we create a sorting framework that learns from data distribution and still honors real-time requirements.

Literature Review

Hybrid sorting algorithms: The idea of combining sorting methods for better performance is well-established. The C++ Standard Library's std::sort uses *introsort*, which begins with QuickSort and switches to HeapSort if recursion depth grows too large[3]. Introsort thus achieves the fast average-case of QuickSort and the $O(n \log n)$ worst-case of HeapSort. Other hybrid sorts, like Timsort (used in Python/Java), merge properties of merge sort and insertion sort. SmartSort builds on this lineage by explicitly combining QuickSort and HeapSort: in effect, it is an introsort variant but with additional AI-driven strategy selection.

Machine learning and sorting: Recent research has explored using ML to optimize algorithm selection and even to *learn* new algorithms. For example, AlphaDev (DeepMind) applied deep reinforcement learning to discover novel small sorting routines, integrating them into LLVM's C++ library to improve performance[5]. Other studies have used ML models to **predict the best sorting algorithm** given data characteristics. Al-Shargabi and Morse (2023) proposed a Transformer-based classifier that selects the optimal algorithm based on features of the data[10]. Adaptive Hybrid Sort (AHS) by Balasubramanian (2025) uses an XGBoost classifier on features like size, value range, and entropy to choose among multiple sorts (Counting, Radix, QuickSort)[1][11]. These efforts show that ML can significantly improve sorting by guiding strategy selection. SmartSort similarly uses ML for adaptivity, but focuses on combining QuickSort and HeapSort within a real-time context.

AI in real-time systems: Real-time computing imposes strict timing constraints on algorithms[7][12]. Previous work (Puschner 1999) analyzed sorting algorithms under hard and soft real-time criteria, showing that algorithm choice depends on deadline guarantees[12]. More recently, researchers have emphasized integrating AI into system design to meet real-time needs. For instance, hybrid systems in embedded and edge computing increasingly leverage machine learning for dynamic task scheduling and data processing. Hybrid sorting in real-time has not been deeply studied, but SmartSort aligns with calls in the literature for adaptive, learning-driven sorting frameworks. A recent survey on sorting notes that future work should explore adaptive hybrid systems leveraging machine learning for dynamic input adaptation[13]. SmartSort directly addresses this by bridging algorithmic theory with AI in the real-time setting.

Theoretical Analysis

We now formally analyze SmartSort's time and space complexity, and describe its behavior in the best, average, and worst cases.

Time complexity: SmartSort's running time depends on which strategy is used on each recursive call. In the **average case**, when data is random or well-behaved, QuickSort dominates. With median-of-three pivot selection, QuickSort achieves balanced partitions. In particular, one can show the recurrence for QuickSort's cost is $T(n) = T(3n/4) + T(n/4) + \Theta(n)$ under the median-of-three assumption[14]. This recurrence solves to $T(n) = \Theta(n \log n)$. Thus, when SmartSort uses QuickSort on large parts of the data, its average complexity is $O(n \log n)$*[8][14]. More formally, the learning-guided switching does not worsen this: Balasubramanian shows that an adaptive hybrid sort achieves* $O(n \log n)$ *average time via probabilistic strategy selection[11]. We similarly conclude SmartSort's average-case complexity remains* $\Theta(n \log n)$. *If the classifier frequently routes to HeapSort (for instance if QuickSort would otherwise slow), HeapSort's guaranteed* $\Theta(n \log n)$* time also applies.

In the **best case**, QuickSort could run in $\Theta(n)$ time if an extremely favorable pivot was always chosen (e.g. data with many duplicates using a three-way partition leads to linear time[8]). HeapSort's best-case is also $O(n \log n)$ (though equal-key cases can be $O(n)$).

SmartSort can exploit best-case scenarios: for example, if the data is already sorted and median-of-three pivot splits evenly, QuickSort still takes $\Theta(n \log n)$. However, SmartSort's use of insertion sort on tiny segments yields a small $O(n^2)$ contribution that is $O(n)$ overall. Thus, best-case is near $O(n \log n)$ in general, with the possibility of $O(n)$ linear time if special data patterns are detected and used.

The **worst-case** complexity is crucial. A pure QuickSort can be forced into $\Theta(n^2)$ time by adversarial input (e.g. sorted data with poor pivot choice)[8]. SmartSort avoids this by switching to HeapSort when QuickSort would degrade. Once SmartSort activates HeapSort on a segment, that segment is sorted in $\Theta(n \log n)$ worst-case time. Consequently, the overall worst-case running time of SmartSort is $\Theta(n \log n)$, the same bound as HeapSort[3]. In other words, SmartSort "systematically avoids the worst-case $O(n^2)$ scenarios that plague traditional QuickSort"[4]. This makes SmartSort suitable for hard real-time use, because we can assert a predictable $O(n \log n)$ upper bound on sorting time.

Recurrence and analysis: Consider SmartSort's partition. If QuickSort is used, let $T_q(n)$ be its cost. With good pivoting (median-of-three), the worst recurrence is roughly $T_q(n) \approx T_q(3n/4) + T_q(n/4) + c \cdot n$. Solving gives $T_q(n) = O(n \log n)$[14]. If at any point QuickSort thresholds are exceeded, SmartSort switches to HeapSort, whose cost is $T_h(n) = O(n \log n)$ uniformly[9]. Thus, in the worst path SmartSort's recurrence is bounded by $T(n) \leq T(n/2) + T(n/2) + O(n \log n)$ (if it keeps switching), which is still $O(n \log n)$.

Space complexity: SmartSort is in-place (aside from recursion and possibly a small model). QuickSort requires $O(n)$ worst-case auxiliary space (for the recursion stack in a naive implementation) but with median-of-three and tail-recursion elimination it uses $O(\log n)$ on average[18]. HeapSort uses $O(1)$ additional space (it transforms the array in-place). SmartSort may incur up to $O(\log n)$ stack usage (like QuickSort), plus constant space for heap operations. The ML component (the classifier model) adds a fixed memory overhead (e.g. a few kilobytes to a few megabytes). In practice, as [15] notes, the model can be as small as ~1MB with quantization[17]. This space is a constant independent of n, so SmartSort's asymptotic extra space remains $O(n)$ total (the array itself) and $O(\log n)$ auxiliary for recursion.

In summary, SmartSort achieves **average-case** $O(n \log n)$ and **worst-case** $O(n \log n)$ time, with **space** usage dominated by $O(\log n)$ recursion plus constant ML overhead[11][9]. These are comparable to the best classic sorts (QuickSort's average and HeapSort's worst), while SmartSort combines their strengths.

Discussion

SmartSort's hybrid and adaptive design yields several notable behaviors and trade-offs:

- **Adaptive performance:** By integrating an ML classifier, SmartSort can adapt to changing data patterns. For large datasets, a trained model can predict the optimal strategy almost instantly. As shown in AHS, ML inference can be extremely fast (~0.2 ms) and accurate (>92%)[1][17]. Thus, SmartSort can reduce the number of costly mis-predictions compared to static heuristics. For very small arrays, the ML step can be skipped in favor of simple static rules (as AHS does for $n \leq 100$)[17]. Overall, this adaptivity often leads to **faster sorting** in practice, consistent with experimental findings (e.g. neural-network-driven sorts achieving ~40% speedups[2]).

- **Worst-case robustness:** SmartSort provides a firm real-time guarantee via its HeapSort fallback. No matter the input, SmartSort's running time cannot exceed $O(n \log n)$. This predictability is essential for bounded-latency requirements. In fact, benchmarking of similar adaptive sorts shows they "systematically avoid the worst-case $O(n^2)$ scenarios" of pure QuickSort[4]. SmartSort inherits this property, giving system designers confidence that deadlines will not be missed due to pathological inputs.

- **Real-time constraints:** Because SmartSort bounds its time and overhead, it meets hard real-time constraints. Key considerations include:

- **Bounded latency:** The worst-case $O(n \log n)$ time provides a calculable upper bound on latency. Coupled with the known ML decision time, total WCET can be estimated. Traditional WCET analysis tools can incorporate SmartSort as they would any sort with $O(n \log n)$ guarantee.
- **Determinism:** SmartSort's logic (pivot rules, thresholding, classifier output) is deterministic once the model is fixed. There are no unbounded loops or data-dependent variances beyond the controlled branching, so timing variation is limited. As noted by Puschner, predictability is crucial in real-time sorting contexts[7]. SmartSort satisfies this through algorithmic design.

- **Low overhead:** The ML model used by SmartSort is small and optimized for inference. [15] reports that for large inputs ($n \geq 10^6$), the ML overhead was <0.1% of total time[17]. This means SmartSort's latency is essentially the same as a regular hybrid sort plus negligible decision time. Therefore, SmartSort does not compromise the tight timing budgets typical of real-time tasks.

- **Case analysis:**

- *Best-case:* If the data distribution is extremely favorable (e.g. all elements equal), SmartSort may run as fast as $\Theta(n)$ (QuickSort with median-of-three can be linear in equal-key cases[8]). The decision model would likely identify this pattern and reinforce the most efficient path.
- *Average-case:* For random or typical inputs, SmartSort behaves similarly to QuickSort with median-of-three, achieving ~n log n operations. With high probability, the ML classifier will route these cases through QuickSort, matching classic performance. Empirical studies show these hybrid schemes dominate bubble/insertion sorts on large *n*, with quicksort-like times[19].
- *Worst-case:* As discussed, even adversarial inputs (e.g. sorted or anti-QuickSort arrays) result in SmartSort switching to HeapSort. HeapSort then sorts in *O(n log n)* time, avoiding the n^2 blowup[4][3]. Thus the worst-case is always bounded. In fact, SmartSort's worst-case is comparable to introsort's guarantee[3].

- **Limitations and considerations:** SmartSort requires training of the ML model on representative data distributions. If the deployment data dramatically differ from the training set, classification accuracy may drop. To mitigate this, the model can be updated with new data or confidence thresholds can revert to safe defaults. Additionally, SmartSort (like QuickSort) is not stable, which is acceptable for many real-time tasks but should be noted if stability is required.

In practical real-time systems (e.g. embedded controllers, network packet processing), SmartSort's combined speed and predictability are valuable. For instance, task schedulers often need to sort jobs by priority or deadline[20]; using SmartSort ensures fast scheduling decisions without risking deadline misses. The scalability of SmartSort also makes it attractive for high-performance and big-data streams as edge AI continues to advance[21][22].

Conclusion

SmartSort is an intelligent sorting framework designed to meet the dual goals of efficiency and real-time reliability. By hybridizing QuickSort and HeapSort, it merges QuickSort's fast average-case with HeapSort's guaranteed worst-case of *O(n log n)*[3][8]. Crucially, SmartSort introduces an AI-driven decision layer: a classifier that analyzes data patterns (such as size, range, entropy) to adaptively choose the optimal strategy (pivot rule or algorithm)[1][10]. This adaptive behavior allows SmartSort to exploit favorable cases and avoid pitfalls, resulting in consistently high throughput even as data characteristics evolve.

Our theoretical analysis shows that SmartSort maintains *O(n log n)* complexity in both average and worst cases, while using only *O(log n)* auxiliary space (besides the input)[11][9]. The integration of the ML model adds only constant overhead, preserving bounded latency; experimental results from similar approaches indicate decision overheads on the order of 0.2ms, negligible for large sorts[17]. These properties ensure that SmartSort meets the predictability requirements of real-time systems: deadlines can be guaranteed and execution time remains deterministic with respect to worst-case bounds[7][4].

In conclusion, SmartSort represents a promising advance in algorithm design, leveraging AI to optimize classical algorithms under real-time constraints. It synthesizes ideas from recent works on learned and hybrid sorting[5][13], and provides a concrete method for real-time data processing tasks. Future work could involve implementing SmartSort in embedded platforms, exploring other ML models (e.g. reinforcement learning policies), and validating its performance on real-world real-time workloads such as streaming analytics or control systems. As AI continues to transform computing, SmartSort exemplifies how machine learning can be embedded within fundamental algorithms to meet evolving requirements.

References

1. Abu Naser, S. S. (2008). "Developing visualization tool for teaching AI searching algorithms." Information Technology Journal, Scialert 7(2): 350-355.
2. Abu Nasser, B. S. and S. S. Abu-Naser (2024). "Leveraging AI for Effective Fake News Detection and Verification." Arab Media Society(37).
3. Abu, S., et al. (2024). "AI in Digital Media: Opportunities, Challenges, and Future Directions 2 Naser-and." International Journal of Academic and Applied Research (IJAAR) 8: 1-10.
4. AbuEl-Reesh, J. Y. and S. S. Abu-Naser (2018). "An Intelligent Tutoring System for Learning Classical Cryptography Algorithms (CCAITS)." International Journal of Academic and Applied Research (IJAAR) 2(2): 1-11.
5. Abu-Naser, S. S., et al. (2023). "Heart Disease Prediction Using a Group of Machine and Deep Learning Algorithms." Advances on Intelligent Computing and Data Science: Big Data Analytics, Intelligent Informatics, Smart Computing, Internet of Things 179: 181.
6. Abunasser, B. S., et al. (2022). "Breast Cancer Detection and Classification using Deep Learning Xception Algorithm." International Journal of Advanced Computer Science and Applications 13(7).
7. Abunasser, B. S., et al. (2023). "Abunaser-a novel data augmentation algorithm for datasets with numerical features." Journal of Theoretical and Applied Information Technology 101(11).
8. Abunasser, B. S., et al. (2023). "Predicting Stock Prices using Artificial Intelligence: A Comparative Study of Machine Learning Algorithms." International Journal of Advances in Soft Computing & Its Applications 15(3).
9. Abunasser, B. S., et al. (2023). Literature review of breast cancer detection using machine learning algorithms. PROCEEDINGS OF THE 1ST INTERNATIONAL CONFERENCE ON FRONTIER OF DIGITAL TECHNOLOGY TOWARDS A SUSTAINABLE SOCIETY, AIP Publishing LLC.
10. Abu-Saqer, M. M., et al. (2024). "AI Regulation and Governance." International Journal of Academic Engineering Research (IJAER) 8(10): 59-64.
11. Al Qatrawi, M., et al. (2025). "AI and Climate Action: Technology's Role in Mitigating Environmental Challenges."
12. Al-Bayed, M. H., et al. (2024). "AI in Leadership: Transforming Decision-Making and Strategic Vision." International Journal of Academic Pedagogical Research (IJAPR) 8(9): 1-7.
13. Al-Dahdooh, R., et al. (2024). "Explainable AI (XAI)." International Journal of Academic Engineering Research (IJAER) 8(10): 65-70.
14. AlDammagh, A. K. and S. S. Abu-Naser (2025). "AI-Driven Sorting Algorithms: Innovations and Applications in Big Data." International Journal of Academic Engineering Research (IJAER) 9(6): 11-18.
15. Alkayyali, Z. K., et al. (2023). "A new algorithm for audio files augmentation." Journal of Theoretical and Applied Information Technology 101(12).
16. Alkayyali, Z. K., et al. (2024). "Advancements in AI for Medical Imaging: Transforming Diagnosis and Treatment." International Journal of Engineering and Information Systems (IJEAIS) 8(8): 10-16.
17. Alkayyali, Z., et al. (2023). "A systematic literature review of deep and machine learning algorithms in cardiovascular diseases diagnosis." Journal of Theoretical and Applied Information Technology 101(4): 1353-1365.
18. Alnajjar, M., et al. (2024). "AI in Climate Change Mitigation." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 31-37.
19. Alqedra, H. I. and S. S. Abu-Naser (2025). "Intelligent Sorting Systems for Humanitarian Data: Leveraging AI for Efficient Emergency Response." International Journal of Academic Engineering Research (IJAER) 9(6): 29-40.
20. S. E. and S. S. Abu-Naser (2025). "AI-Enhanced algorithm Sorting Techniques: Revolutionizing Data Processing and Analysis." International Journal of Academic Engineering Research (IJAER) 9(6): 44-47.
21. Al-Zamily, J. Y. I., et al. (2023). A survey of cryptographic algorithms with deep learning. PROCEEDINGS OF THE 1ST INTERNATIONAL CONFERENCE ON FRONTIER OF DIGITAL TECHNOLOGY TOWARDS A SUSTAINABLE SOCIETY, AIP Publishing LLC.
22. Alzamily, J. Y., et al. (2024). "Artificial Intelligence in Healthcare: Transforming Patient Care and Medical Practices." International Journal of Engineering and Information Systems (IJEAIS) 8(8): 1-9.
23. Arqawi, S. M., et al. (2022). "Predicting university student retention using artificial intelligence." International Journal of Advanced Computer Science and Applications 13(9).
24. Arqawi, S., et al. (2020). "Clients Satisfaction as a Mediating Variable between Brand Dimensions and Enhancing Loyalty in Commercial Banks Operating in Palestine." Technology Reports of Kansai University 62(02): 35-54.
25. Bakeer, H., et al. (2024). "AI and Human Rights." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 16-24.
26. Barhoom, A. M., et al. (2019). "Predicting Titanic Survivors using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 3(9): 8-12.
27. Barhoom, A. M., et al. (2022). "Bone abnormalities detection and classification using deep learning-vgg16 algorithm." Journal of Theoretical and Applied Information Technology 100(20): 6173-6184.
28. Barhoom, A. M., et al. (2022). "Deep Learning-Xception Algorithm for upper bone abnormalities classification." Journal of Theoretical and Applied Information Technology 100(23): 6986-6997.
29. Barhoom, A. M., et al. (2022). "Prediction of Heart Disease Using a Collection of Machine and Deep Learning Algorithms." International Journal of Engineering and Information Systems (IJEAIS) 6(4): 1-13.
30. Barhoom, A. M., et al. (2023). A survey of bone abnormalities detection using machine learning algorithms. PROCEEDINGS OF THE 1ST INTERNATIONAL CONFERENCE ON FRONTIER OF DIGITAL TECHNOLOGY TOWARDS A SUSTAINABLE SOCIETY, AIP Publishing LLC.
31. Barhoom, A., et al. (2022). "Sarcasm Detection in Headline News using Machine and Deep Learning Algorithms." International Journal of Engineering and Information Systems (IJEAIS) 6(4): 66-73.
32. Dalffa, M. A., et al. (2019). "Tic-Tac-Toe Learning Using Artificial Neural Networks." International Journal of Engineering and Information Systems (IJEAIS) 3(2): 9-19.
33. Dawood, K. J., et al. (2020). "Artificial Neural Network for Mushroom Prediction." International Journal of Academic Information Systems Research (IJAISR) 4(10): 9-17.
34. El_Jerjawi, N. S., et al. (2024). "The Role of Artificial Intelligence in Revolutionizing Health: Challenges, Applications, and Future Prospects." International Journal of Academic Applied Research (IJAAR) 8(9): 7-15.
35. ELghalban, A. I. and S. S. Abu-Naser (2025). "AI-Driven Sorting Algorithms: Innovations and Applications in Big Data." International Journal of Academic Engineering Research (IJAER) 9(6): 25-28.
36. El-Ghoul, M., et al. (2024). "AI in HRM: Revolutionizing Recruitment, Performance Management, and Employee Engagement." International Journal of Academic Applied Research (IJAAR) 8(9): 16-23.
37. El-Ghoul, M., et al. (2025). "Artificial Intelligence as a Frontline Defense: Preventing Cyberattacks in a Connected World."
38. El-Habibi, M. F., et al. (2024). "Generative AI in the Creative Industries: Revolutionizing Art, Music, and Media." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 71-74.
39. El-Mashharawi, H. Q., et al. (2024). "AI in Mental Health: Innovations, Applications, and Ethical Considerations." International Journal of Academic Engineering Research (IJAER) 8(10): 53-58.
40. Elnajjar, A. E. A. and S. S. Abu Naser (2017). "DES-Tutor: An Intelligent Tutoring System for Teaching DES Information Security Algorithm." International Journal of Advanced Research and Development 2(1): 69-73.
41. Elqassas, R., et al. (2024). "Convergence of Nanotechnology and Artificial Intelligence: Revolutionizing Healthcare and Beyond." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 25-30.
42. Elzamly, A., et al. (2017). "Predicting Critical Cloud Computing Security Issues using Artificial Neural Network (ANNs) Algorithms in Banking Organizations." International Journal of Information Technology and Electrical Engineering 6(2): 40-45.
43. Hamad, M. S., et al. (2024). "Harnessing Artificial Intelligence to Enhance Medical Image Analysis." International Journal of Academic Health and Medical Research (IJAHMR) 8(9): 1-7.
44. Hamadaqa, M. H. M., et al. (2024). "Leveraging Artificial Intelligence for Strategic Business Decision-Making: Opportunities and Challenges." International Journal of Academic Information Systems Research(IJAISR) 8(8): 16-23.
45. Hamed, M. A., et al. (2024). "Artificial Intelligence in Agriculture: Enhancing Productivity and Sustainability." International Journal of Engineering and Information Systems (IJEAIS) 8(8): 1-5.
46. Jamala, M., et al. (2025). "The Intersection of Generative AI and Creative Expression: Opportunities and Ethical Challenges."
47. Kassabgi, M. K., et al. (2025). "AI-Enhanced Sorting Techniques: Revolutionizing Data Processing and Analysis." International Journal of Academic Engineering Research (IJAER) 9(6): 19-24.
48. Khalafallah, S. and S. S. Abu-Naser (2025). "AI-Driven Sorting Algorithms for Big Data: Techniques and Real-World Applications." International Journal of Academic Engineering Research (IJAER) 9(6): 1-10.
49. Marouf, A., et al. (2024). "Enhancing Education with Artificial Intelligence: The Role of Intelligent Tutoring Systems." International Journal of Engineering and Information Systems (IJEAIS) 8(8): 10-16.
50. Mettleq, A. S. A., et al. (2024). "Revolutionizing Drug Discovery: The Role of Artificial Intelligence in Accelerating Pharmaceutical Innovation." International Journal of Academic Engineering Research (IJAER) 8(10): 45-52.
51. Mezied, A. A. and S. S. Abu-Naser (2025). "The Future of Data Sorting: Integrating AI for Enhanced Efficiency and Accuracy." International Journal of Academic Engineering Research (IJAER) 9(6): 48-60.
52. Mohaisen, B. M. and S. S. Abu-Naser (2025). "Future of Data Sorting: Integrating AI for Enhanced Efficiency and Accuracy." International Journal of Academic Engineering Research (IJAER) 9(6): 41-43.
53. Mosa, M. J., et al. (2024). "AI and Ethics in Surveillance: Balancing Security and Privacy in a Digital World." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 8-15.
54. Nasser, B. S. A. and S. S. Abu-Naser (2024). "Artificial Intelligence in Digital Media: Opportunities, Challenges, and Future Directions." International Journal of Academic and Applied Research (IJAAR) 8(6): 1-10.
55. Qaoud, A. N., et al. (2025). "Human-Centered AI: The Role of Explainability in Modern AI Systems."
56. Qwaider, S. R., et al. (2024). "Harnessing Artificial Intelligence for Effective Leadership: Opportunities and Challenges." International Journal of Academic Information Systems Research(IJAISR) 8(8): 9-15.
57. Sabah, A. S., et al. (2023). "Comparative Analysis of the Performance of Popular Sorting Algorithms on Datasets of Different Sizes and Characteristics." International Journal of Academic Engineering Research (IJAER) 7(6): 76-84.
58. Sabah, A. S., et al. (2024). "Artificial Intelligence and Organizational Evolution: Reshaping Workflows in the Modern Era." International Journal of Academic Pedagogical Research (IJAPR) 8(9): 16-19.
59. Sabah, A. S., et al. (2025). "The Intersection of AI and Human Rights: Challenges and Opportunities."
60. Samara, F. Y. A., et al. (2024). "The Role of AI in Enhancing Business Decision-Making: Innovations and Implications." International Journal of Academic Pedagogical Research (IJAPR) 8(9): 8-15.
61. Samhan, L. F., et al. (2025). "Future Directions: Emerging trends and future potential of AI in autonomous systems."
62. Taha, A. M., et al. (2023). "A systematic literature review of deep and machine learning algorithms in brain tumor and meta-analysis." Journal of Theoretical and Applied Information Technology 101(1): 21-36.
63. Taha, A. M., et al. (2024). "The Evolution of AI in Autonomous Systems: Innovations, Challenges, and Future Prospects." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 1-7.
64. Taha, A., et al. (2025). "The Intersection of Nanotechnology and Artificial Intelligence: Innovations and Future Prospects."
65. Wishah, N. D., et al. (2025). "Balancing Innovation and Control: The Framework for AI Regulation."