Computer Science Review 59 (2026) 100850

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Time-sensitive data analytics: A survey of anytime techniques, applications and challenges

Jagat Sesh Challa, Aarti, Navneet Goyal, Poonam Goyal *

Department of Computer Science & Information Systems, Pilani Campus, Birla Institute of Technology and Science, Pilani, India

ARTICLE INFO

Keywords:
Data analytics
Classification
Clustering
Anomaly detection
Frequent itemset mining
Anytime algorithms

ABSTRACT

In the era of big data and real-time analytics, there is a growing demand for fast, adaptive, and efficient techniques for data analytics that are not only accurate but also responsive and adaptable to dynamic environments. Anytime algorithms have gained significant attention in data analytics due to their ability to provide approximate results at any point in time (which improves over time), making them highly suitable for quick decision-making. Anytime algorithms, which can trade computational time for quality of results, are increasingly critical for applications requiring rapid, adaptive insights. They are widely used in stock market analysis, fraud detection, sentiment analysis, weather forecasting, etc. To the best of our knowledge, there is no literature survey of research papers on anytime algorithms that comprehensively reviews the approaches, classifies them and highlights the open research issues. This paper provides a comprehensive survey of anytime algorithms tailored for data analytics over large datasets while emphasizing their application in time-sensitive decision-making environments. We examine the algorithmic foundations and the state-of-the-art anytime approaches across various data analytics tasks, including classification, clustering and frequent itemset mining. Qualitative analysis has also been presented for each algorithm described in this paper based on key aspects such as interruptibility, resource adaptiveness, and solution quality under constrained conditions. This survey also highlights the latest advancements and emerging research trends, providing insights into how anytime algorithms can be further developed to meet the demands of complex and dynamic environments.

Contents

* Corresponding author.
E-mail addresses: jagatsesh@pilani.bits-pilani.ac.in (J.S. Challa), p20210419@pilani.bits-pilani.ac.in (Aarti), goel@pilani.bits-pilani.ac.in (N. Goyal), poonam@pilani.bits-pilani.ac.in (P. Goyal).

https://doi.org/10.1016/j.cosrev.2025.100850
Received 14 June 2025; Received in revised form 25 September 2025; Accepted 25 October 2025
Available online 30 October 2025

1. Introduction

Data analytics refers to the process of discovering useful patterns, trends, and relationships within large data by using a variety of techniques from statistics, machine learning, and artificial intelligence. It involves analyzing data to extract meaningful insights and transforming them into actionable knowledge [1]. With the rapid growth of data generated by businesses, social media, and IoT devices, data analytics has become an essential tool for organizations seeking to gain a competitive advantage through data-driven decision-making.

There are various data analytics tasks including — classification [2,3], clustering [4], anomaly detection [5], association rule mining [6], and sequence mining [1]. Classification assigns objects to one of several predefined categories, which is useful in many diverse applications that include — detecting spam email messages based on the message header and content [1], categorizing cells as malignant or benign based on the results of MRI scans [1], classifying galaxies based on their shapes [1], Fraud Detection in credit card transactions [6], etc. Cluster analysis seeks to find groups of closely related objects so that the objects that belong to the same cluster are more similar to each other than the objects that belong to other clusters. A few applications of clustering include — grouping sets of related customers [1], finding ocean areas that significantly impact the earth's climate [7], grouping astronomical galaxies [8], etc. Association analysis discovers patterns that describe strongly associated features in the data. The applications of association analysis include — finding groups of genes that have related functionality [9], identifying web pages that are accessed together [10], identifying the patterns of frequently purchased items at retail stores [11], etc.

1.1. Challenges faced by traditional algorithms

Traditional data analytics techniques often struggle to process large-scale data, especially in environments where the computational resources and the processing time are constrained. These include issues with scalability, processing speed, adaptability to complex structures, data processing capabilities, and balancing trade-offs between accuracy and computational efficiency, particularly in resource-constrained environments. They are often inadequate for handling variable inter-arrival rates of data. They run on their own limited speed handling capability and are not flexible enough to handle variable speeds. If they were to be used at speeds beyond their maximum capability, they would have to either process sampled data or buffer unlimited data and eventually get stuck [12]. For example, while traditional SVMs require fixed kernel evaluations, anytime SVMs [13] reduce evaluations dynamically, improving efficiency in resource-constrained environments. Table 1 shows the comparison between budget/non-anytime vs anytime algorithms for large-scale static data.

In such scenarios, *Anytime algorithms* [14–17] come to the rescue, wherein they can provide incremental insights from the analysis that improve with an increase in the allocated resources, such as computational resources and time. Anytime algorithms are a class of algorithms that offer a trade-off between the constraints on available resources and the quality of results. They can be interrupted at any time before their completion and provide intermediate, valid approximate results. The quality of that result can be improved with an increase in resources allocated. Resources could be processing time allowance, computational resources, etc. This behavior of an anytime algorithm is depicted in Fig. 1, where we can see that the accuracy of the result improves with an increase in processing time allowance.

There are many application domains where anytime algorithms are applicable in decision-making. For example, consider the application of stock market analysis to predict trends from large-scale data [14]. Stock markets generate vast amounts of real-time data, including stock prices and trading volumes, which keep on varying from time to time. Anytime algorithms can process this data incrementally, providing preliminary insights quickly if interrupted and refining predictions as more resources become available. This kind of algorithm benefits both a short-term stock investor who is interested in quick results, as well as a long-term stock investor who is willing to wait until a more refined and accurate result is computed. Other application domains where anytime algorithms are useful include — sentiment analysis in social media analytics, speech recognition in NLP [18], object recognition from images and videos [19], fraud detection in credit card transactions [20], path planning in robotics [21], etc. The development of anytime algorithms reflects a growing need for efficient and timely computation of solutions. By allowing for progressive refinement and early access to results, these algorithms empower users to make informed decisions quickly while still benefiting from more comprehensive analysis over time. Anytime algorithms are particularly useful for handling large and dynamic data, where traditional methods may struggle to keep up with the volume and speed at which the data is generated. By providing approximate results at any point in time, anytime algorithms enable rapid insights and adaptability, making them an attractive solution for a wide range of data-driven applications.

Anytime algorithms can be categorized as - *contract* and *interruptible*. Both types can provide useful insights when constrained by resources. They differ in how they manage allocated resources and provide useful results during execution. A contract algorithm is one that gets its resources (time, computational hardware, etc.) allocated beforehand, and the algorithm executes within those allotted resources to produce the best possible result. Contract algorithms, if interrupted before the contract ends, don' t guarantee a valid result. An interruptible algorithm, on the other hand, is designed to give a result at any point of interruption during its execution. An interruptible algorithm is one whose resource allocation is not given in advance. If interrupted, it will return the best solution it has found so far, and if allowed to continue, it will refine its result further. For instance, in a fraud detection system, if the algorithm is interrupted before it finishes, it can produce quick fraud alerts, and if more processing time is given, it can refine the accuracy of these alerts as more data gets analyzed. This feature makes interruptible algorithms highly flexible and suitable for environments where computational resources or time availability are uncertain.

1.2. Anytime algorithms in data analytics

In the context of data analytics, anytime algorithms can be applied to two broader scenarios - *mining large databases* and *mining data*

Table 1
Comparison between non-anytime and anytime algorithms in data analytics.

Aspect	Non-anytime algorithms	Anytime algorithms
Execution model	Run to completion before producing any output.	Can be interrupted at any point to provide the best-so-far result.
Result availability	Only available after full processing.	Progressive refinement; output improves with an increase in processing time.
Adaptability to time constraints	Fixed runtime	Adapts to varying time allowances.
Suitable for large datasets	Requires full dataset processing before producing the usable results.	Can work progressively, giving early, approximate results.
Handling resource constraints	Fixed computational budget	Adjusts processing based on available resources (memory, time).
Scalability	Requires more hardware	Provides intermediate results, and refines over time.
Use in real-time decision-making	Cannot return intermediate insights before completion.	Well-suited for real-time and streaming environments where quick decisions are needed.
Quality of intermediate results	No intermediate result; must wait for the final result.	Quality improves with an increase in time; early results may be coarse but usable.

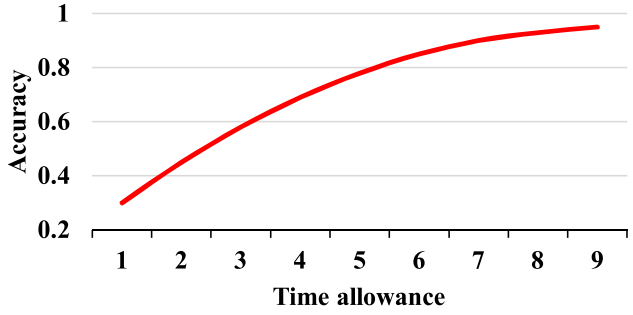

Fig. 1. Characteristic of an anytime algorithm [12].

streams. In this survey paper, we focus on anytime algorithms for mining large databases.

Anytime algorithms for large datasets produce multiple results of various approximations, whose accuracy increases with an increase in processing time allowance. This gives the user flexibility in the trade-off between computational resources and the accuracy of the mining results. Anytime algorithms for data analytics include- induction-tree based [22,23], kernel-based [13], ensemble-based [24–26], neural networks based [27–29], nearest-neighbors based [30], probability estimation based [31,32], density-based [33–36], frequent itemset mining [14,20], etc.

Similarly, if we categorize the algorithms on the basis of contract and interruptible, anytime decision trees [22,23], anytime frequent itemset mining [14], etc., come under contract algorithms. Anytime interruptible decision trees [37], anytime nearest neighbors [16,30,38], anytime bayesian classifiers [39], anytime set-wise classification [40], etc. are the interruptible algorithms.

To measure the performance of anytime algorithms for large datasets, the focus is on their ability to provide progressively improved results when more resources (processing time and computational resources) become available. The quality of the solution (accuracy) produced under varied (generally improving) constrained conditions is used to evaluate the effectiveness of an anytime algorithm. Fig. 2 describes the framework for anytime algorithms.

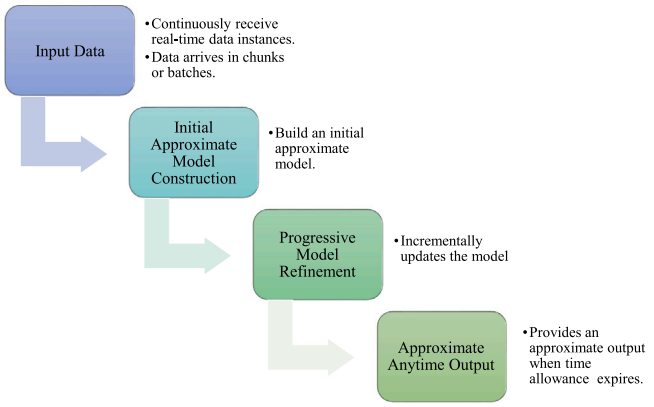

Fig. 2. Framework for anytime algorithms.

1.3. Metrics for performance evaluation of anytime algorithms

To evaluate the performance of anytime algorithms, particularly for large datasets, metrics are required that capture their ability to provide progressively improved solutions as more computational resources (e.g., processing time, memory) become available. Unlike traditional algorithms, which return a single final result after completing the execution, however, the performance of the anytime algorithms can be evaluated based on the quality of the intermediate results and the speed at which they improve over time. The following metrics are recommended to systematically assess their performance:

- **Initial Solution Quality under Interruption**: Measures the quality of the first available intermediate solution when the algorithm is interrupted early.
- **Time-Computational Efficiency Trade-off**: Evaluates how efficiently an anytime algorithm transforms available computational resources (e.g., CPU time, memory usage, etc.) into quality improvements such as improved accuracy, F1-score, reduced classification cost, or higher clustering purity—in its intermediate and final solutions with an increase in processing time allowance (refer to Fig. 1). Better computational efficiency means that even if the algorithm is interrupted early, it will deliver more useful results.
- **Scalability**: Measures how well an algorithm maintains its anytime properties with an increase in size or complexity of data or stream speed.

1.3.1. Characteristics of anytime algorithms

Fig. 3 showcases the key characteristics of anytime algorithms (i) *Interruptibility*, (ii) *Resource Adaptiveness*, (iii) *Incremental Learning*, and (iv) *Scalability*.

(i) *Interruptibility*: Anytime algorithms can be stopped at any point in time and still return the best possible approximate result achieved so far. This is imperative in real-time or resource-constrained environments where computation may need to be halted unexpectedly.

(ii) *Resource Adaptiveness*: Anytime algorithms are designed to dynamically adjust their computation based on available resources like time, memory, or computational power. When more resources are available, the algorithm refines its solution and provides a more accurate result.

(iii) *Incremental Learning*: The model can update itself incrementally as new data objects arrive, without needing to retrain from scratch. This is crucial for adapting to evolving data patterns over time.

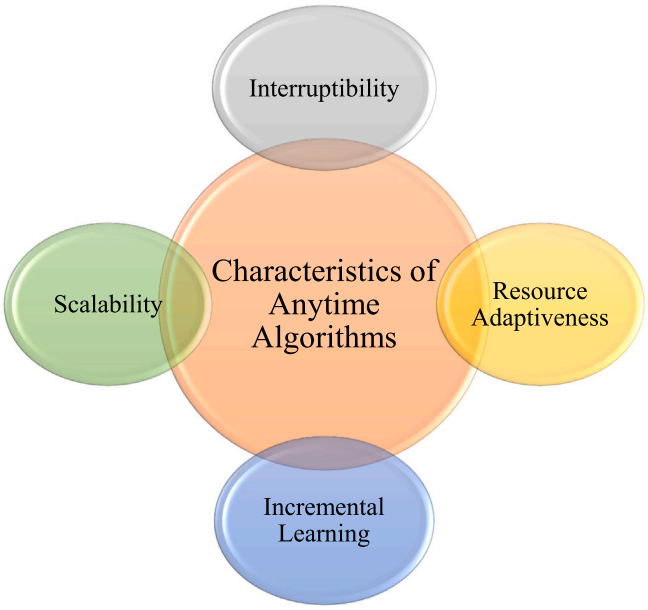

Fig. 3. Key characteristics of anytime algorithms.

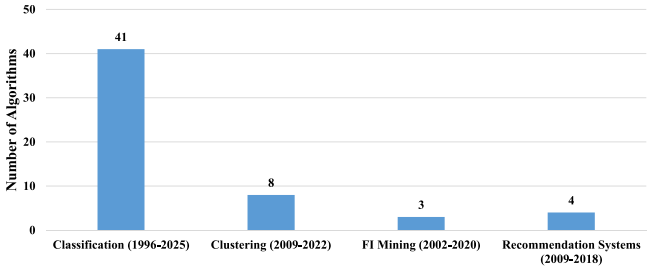

Fig. 4. Overview of anytime algorithms for data analytics in large data.

(iv) *Scalability*: The algorithm can handle increasing volumes of data without a significant drop in performance. It remains efficient even as data streams become larger and faster.

1.4. Our contribution

In this paper, we provide an extensive literature survey of anytime algorithms by considering 56 research articles to examine all relevant research accomplished in the field of data analytics for large datasets. Fig. 4 presents an overview of the number of anytime algorithms developed for different data analytics tasks applied to large datasets, along with the span of years in which these algorithms were proposed.

Classification (1996–2025) shows the highest number of anytime algorithm contributions, with 41 algorithms identified over nearly three decades. 8 methods were proposed for clustering from 2009 to 2022, 3 methods exist for frequent itemset mining (2002–2020), and 4 methods were proposed for recommendation systems (RS) (2009–2018). The relatively low number reflects the computational complexity of Frequent Itemset mining. The survey based on the various types of anytime algorithms across different data analytics tasks is also analyzed in terms of publications per year and key factors of anytime algorithms. Fig. 5 shows the number of publications on anytime algorithms published per year for different data analytics tasks from 1996 to 2025. The trend highlights a steady growth in research interest, with a significant rise after 2010, indicating the increasing relevance of anytime algorithms in time-sensitive and large-scale data analytics tasks. Table 4, 5 summarizes the anytime algorithms for different analytics tasks that are

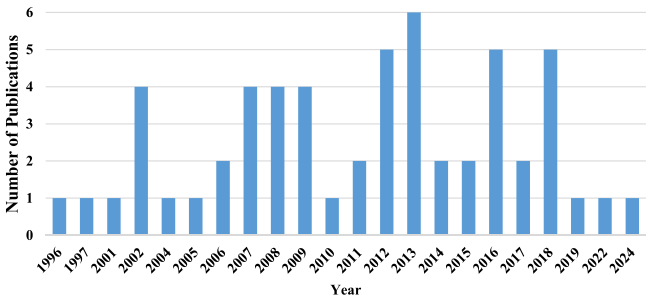

Fig. 5. Year-wise summary of publications in anytime data analytics.

useful for handling large-scale data based on different machine learning models and highlighting the specific features of each algorithm.

This survey essentially addresses the following research questions:

- **RQ1:** What are the key findings of this extensive literature survey on anytime algorithms for data analytics in the context of large datasets?
- **RQ2:** What are the fundamental principles that underpin the design of anytime algorithms for various tasks of data analytics?
- **RQ3:** In what way have the anytime algorithms for data analytics evolved over time, and what are the major improvements over previous approaches in terms of theoretical validation?
- **RQ4:** What are the significant research challenges identified in the area of anytime algorithms for data analytics? What are the future research opportunities to extend the capabilities of these algorithms or propose new algorithms?
- **RQ5:** How can parallelization techniques enhance the scalability and efficiency of anytime algorithms for data analytics?

The rest of the paper is organized as follows: Section 2 explains various methods proposed for anytime classification. Section 3 discusses different approaches developed for anytime clustering. Section 4 describes the methods introduced for frequent itemset mining. Section 5 discusses the algorithms designed for recommendation systems. Section 6 gives a brief insight into the anytime algorithms for data analytics in stream environments. Section 7 discusses the limitations of the survey. Section 8 discusses the significant research issues based on the survey conducted and future directions for research. Finally, Section 9 summarizes the methods presented in the previous sections and concludes the manuscript.

2. Anytime classification

We give a comprehensive survey of different types of anytime classification algorithms that were proposed in the literature, including traditional classification algorithms, active learning based algorithms, vision applications-based algorithms and a few others. The anytime traditional classification algorithms for large datasets are summarized in Fig. 6, while Fig. 7 presents application-based anytime classification algorithms. Table 2 presents a summary of datasets used to evaluate various anytime classification algorithms, providing the best possible results. The datasets range from small synthetic datasets (e.g., XOR-10, Multi-XOR, Synthetic with sizes under 1000 instances) to large-scale real-world datasets such as ImageNet (1.2M instances), Letter (20,000 instances), Munin1 (5 million entries), and Skin Nonskin (245,057 instances). Classification accuracy is the most commonly reported metric, though other performance indicators like F1-score, AUC, regret, log-likelihood, and Kullback score are also considered, depending on the nature of the application.

Decision tree-based anytime approaches such as ID3-k, LSID3, and ACT were evaluated on symbolic datasets like Tic-tac-toe and XOR-10, demonstrating high accuracy (up to 100%) and effective interruptibility

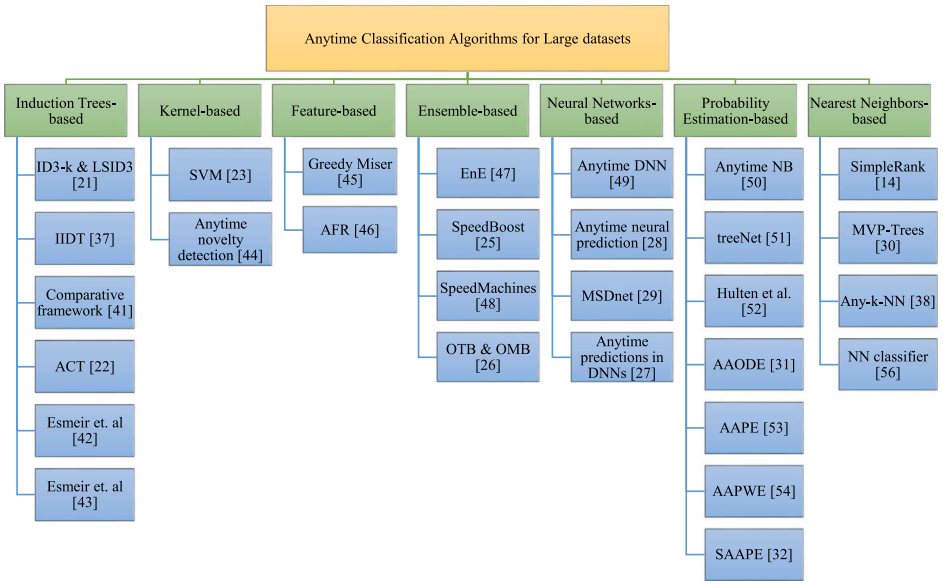

Fig. 6. Categorization of anytime traditional classification algorithms for large data.

Table 2
Datasets used for evaluation of anytime classification algorithms that achieve the highest performance.

Algorithm	Dataset characteristics	Results
ID3-k & LSID3 [22]	Name: Tic-tac-toe, Size: 958	Accuracy: 88%
IIDT [37]	Name: XOR-10, Size: 10 000	Accuracy: 100%
Esmeir et al. [41]	Name: XOR-10, Size: 10 000	Accuracy: 100%
ACT [23]	Name: XOR-10, Size: 10 000	Accuracy: >80%
Esmeir et al. [42]	Name: Multi-XOR, Size: 200	Accuracy: >98%
Esmeir et al. [43]	Name: Multi-XOR, Size: 200	Accuracy: >78%
DeCoste et al. [13]	Name: Sonar, Size: 208	Accuracy: >90%
Sofman et al. [44]	Name: Outdoor Mobile Robot, Size: 6000	True Positive Rate: >90%
Greedy Miser [45]	Name: Scene-15, Size : 4485	Accuracy: >80%
AFR [46]	Name: Synthetic, Size: 1000	Accuracy: >75%
EnE [47]	Name: Switchboard Corpus, Size: 2500	Accuracy: >45%
SpeedBoost [25]	Name: Pendigits, Size : 7494	Accuracy: 98%
SpeedMachines [48]	Name: Stanford Background, Size: 715	Accuracy: 80%
OTB & OMB [26]	Name: Freiburg EEG, Size: 79	No. of seizures detected: >65
Anytime DNN [49]	Name: ImageNet, Size: 1.2M	Accuracy: 92.9%
Lee et al. [28]	Name: ImageNet, Size: 1.2M	Accuracy: >78%
MSDnet [29]	Name: ImageNet, Size: 1.2M	Accuracy: 75%
Hu et al. [27]	Name: ImageNet, Size: 1.2M	Accuracy: 76%
Anytime NB [50]	Name: Synthetic, Size: 21	Kullback score: 0.1
treeNet [51]	Name: Synthetic, Size: 22	Error rate : 0
Hulten et al. [52]	Name: Munin1, Size: 5M	Log-likelihoods: 38.417
AAODE [31]	Name: Pendigits, Size: 7494	Error rate : 0.6
AAPE [53]	Name: Pendigits, Size: 7494	Error rate : 0.86
AAPWE [54]	Name: Pendigits, Size: 7494	Error rate : 0.215
SAAPE [32]	Name: Pendigits, Size: 7494	Error rate : 0.185
SimpleRank [16]	Name: Letter, Size: 20 000	Accuracy: 100%
MVP-Trees [30]	Name: Image, Size: 10 221	Accuracy: 98%
Any-*k*-NN [38]	Name: Skin Nonskin, Size: 245 057	F1-score: 94%
NN classifier [55]	Name: Letter, Size: 20 000	Accuracy: 100%
IEThresh [56]	Name: RTE, Size: 100	Accuracy: 92%
Tomanek et al. [57]	Name: MUC7T, Size: 3113	F1-score: >85%
AAL [58]	Name: MUC7T, Size: 3113	F1-score: 88%
Ramirez et al. [59]	Name: IMDB, Size: 26 784	AUC:0.79
Karayev et al. [19]	Name: PASCAL VOC, Size: 9963	Average Precision: 66%
Anytime Scenes [60]	Name: Scene-15, Size: 4485	Accuracy: 80%
Liu et al. [61]	Name: Leuven Street Scenes, Size: 293	Accuracy: 90%
ICF [62]	Name: Leuven Street Scenes, Size: 293	Accuracy: 89.55%
AFS [63]	Name: Iris, Size: 150	Error rate : 2.0
Schlobach et al. [64]	Name: DICE, Size: 4859	Recall: 90%
APM [65]	Name: Dynamic Pricing, Size: 5 × 5	Regret: 0
Viet et al. [66]	Name: TwoPat, Size: 5000	Accuracy: 97.72%
Aria et al. [67]	Name: IMDB, Size: 26 784	Confidence: 95%
Ben-Shimon [68]	Name: MovieLens, Size: 1M	AUC:80.34
Ben-Shimon et al. [69]	Name: MovieLens, Size: 1M	Precision: 0.5
KARPET [70]	Name: DBLP, Size: 5M	Run time: 2–5 ms

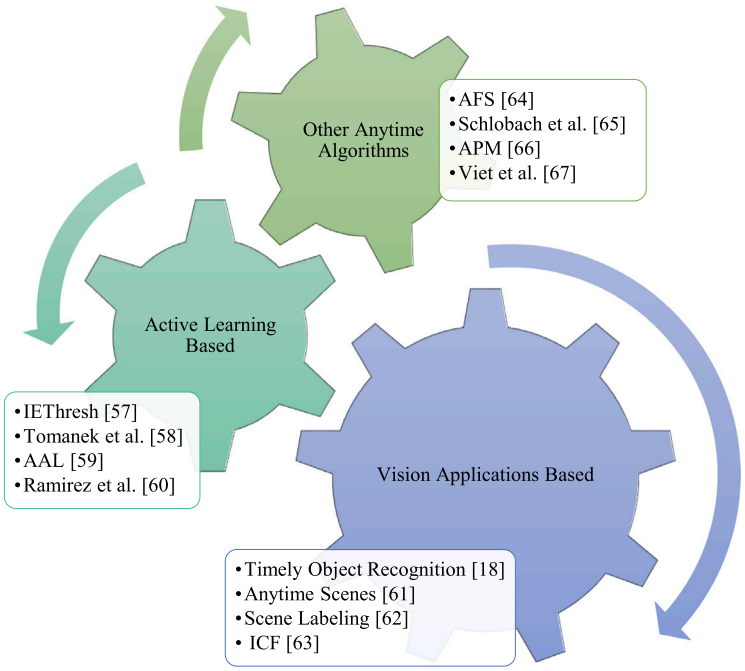

Fig. 7. Categorization of anytime application-based classification algorithms for large data.

in constrained scenarios. More recent advancements leverage deep neural networks (e.g., Anytime DNN, MSDnet, and Hu et al.'s method), achieving competitive accuracy (above 75%–92%) on complex image datasets such as ImageNet. Similarly, algorithms like SpeedBoost, AAPE variants, and SimpleRank show high accuracy or low error rates on digit classification benchmark datasets such as Pendigits and Letter, reinforcing the reliability of decision tree ensembles and probabilistic classifiers for real-time learning. Several approaches address domain-specific challenges: algorithms like Sofman et al. and OTB & OMB are applied to robotics and biomedical datasets (e.g., Outdoor Mobile Robot, Freiburg EEG), while others like AAL, AAPE, and ICF are tested on street scenes, IMDB, and MUC7T, reflecting real-world utility in fields such as autonomous navigation, emotion detection, and natural language processing, also illustrating the broad applicability of anytime classification. Notably, methods like Any-k-NN and AAL focus on real-time performance with high F1-scores and are suited for data stream processing and active learning scenarios. The table demonstrates that anytime classification algorithms maintain competitive performance while offering flexibility in computational resources and decision timing, making them valuable for time-constrained or dynamic environments. Overall, the table highlights the diversity in dataset characteristics and the versatility of anytime classification algorithms in achieving reliable results across different data analytics models.

Now, let us delve into the insights of these algorithms.

2.1. Induction tree-based anytime algorithms

Traditional decision tree induction methods, such as CART, ID3, and C4.5, rely on local heuristics to produce smaller trees but often fail to achieve globally optimal solutions. To address this limitation, anytime decision tree induction algorithms were proposed, which produce a better, higher-quality decision tree as additional processing time is allocated. These algorithms also account for testing and misclassification costs, leading to the induction of cost-sensitive decision trees that aim to minimize the total classification cost while maximizing accuracy.

Esmeir et al. [22] introduce lookahead-based algorithms for anytime induction of decision trees, specifically ID3-k and LSID3. These algorithms are capable of constructing better decision trees when more time becomes available, thereby improving upon greedy algorithms. Essentially, during the tree construction, they predict the profitability of a node split by estimating its impact on deeper node descendants, giving better quality decision trees. Furthermore, the LSID3 algorithm is a contract algorithm (apriori resource allocation) allowing for the utilization of additional resources (time) to construct more refined decision trees. However, the approach has higher computational overhead, limiting scalability to large datasets.

To overcome the limitations of contract algorithms, Esmeir et al. [37] developed interruptible anytime decision trees. Unlike contract-based approaches, these algorithms do not require prior resource allocation (execution time), can be interrupted at any time during execution and are capable of providing improved solutions. The first proposed algorithm involves the conversion of the LSID3 contract algorithm into an interruptible variant. The second algorithm is referred to as IIDT (Interruptible Induction of Decision Trees), which iteratively replaces sub-trees of the current tree with sub-trees generated using higher resource allocations, with the expectation of yielding better results. This approach incurs high recomputation overhead during subtree replacement and is still limited in large-scale, high-dimensional datasets. Esmeir et al. [41] introduced a comprehensive framework for anytime induction of decision trees, enabling hard-to-learn concepts to leverage additional computational resources to produce better hypotheses and exploit larger time budgets. This work provides a comparative analysis of anytime decision tree models, including LSID3 and IIDT, in relation to traditional decision tree models that use bagging, skewing, and GATree. The study concludes that, under scenarios where resource constraints are not a limiting factor, LSID3 and IIDT emerge as the top-performing models, effectively utilizing the available resources to produce more accurate and robust decision trees.

Esmeir et al. [23] introduced a sampling-based method where the cost of each sub-tree, generated by candidate splits, is estimated using Anytime Cost-Sensitive Trees (*ACT*) method. In this method, the split that minimizes the overall cost is favored in the tree construction process. This method works as a contract anytime algorithm, which allows for trading learning time to achieve higher classification accuracy. By leveraging additional time resources, the algorithm obtains

better estimates of the different candidate splits, resulting in more cost-efficient decision trees. Due to the contract-based approach, it may overfit cost estimates with small budgets.

Esmeir et al. [42] introduced an approach which works in environments where computation time can be traded for both test and misclassification costs. It is built on LSID3, which is not designed to minimize test and misclassification costs. The proposed approach leverages additional time and computational resources to generate decision trees with lower costs, demonstrating good anytime behavior with diminishing returns. LSID3 uses SID3 to bias the samples towards small trees; however, ACT does towards low-cost trees. Cost optimization in decision tree induction depends heavily on the accuracy of the underlying cost model, and unrealistic cost assumptions may yield suboptimal trees. *TATA* (Tree-classification AT Anycost) is another anytime framework introduced by Esmeir et al. [43] that produces anytime decision-tree classifiers. It allows for dynamic allocation of learning time and can be configured to work under various budget schemes for classification. This flexibility enables TATA to either produce classifiers with fixed time or cost limits or to continue classifying until interrupted. Notably, this method leverages additional learning time to produce anycost classifiers by forming larger samples, which improves tree-utility estimations. However, requiring many samples for high accuracy increases the runtime in large datasets.

2.2. Kernel-based anytime algorithms

Support vector machines (SVMs) classify objects using support vectors and use multiplication of objects and support vectors to perform kernel computations. In traditional SVM models, the classification cost of a query object is the same irrespective of its difficulty level. The same number of kernel evaluations (dot products between objects and support vectors) is used for every data object. However, we can reduce the number of such kernel evaluations for a few data objects to enhance SVM's performance.

DeCoste [13] presented a computational geometry-based method that reduces the classification cost and the number of kernel computations used in Support Vector Machines (SVMs), making it more efficient for large datasets. Traditional SVM classification requires a substantial number of kernel computations, especially for large datasets, because it applies a uniform number of kernel computations to all objects, which can be computationally expensive. However, this approach produces the same final classification results as a traditional SVM but with fewer kernel computations for "easier" cases. In this approach, the classification cost is proportional to the difficulty level of each object: "easy" points (those lying far from the decision boundary with large margins) are classified with fewer kernel computations, while "hard" points (those near the margin) require more evaluations rather than applying a uniform cost across all objects. The method is designed to work in an anytime fashion, making it particularly useful for real-time or resource-constrained environments. This approach reduces the number of steps(k) and performs efficiently with reduced error rates as k increases. However, margin-based computation prioritization may still incur high costs for borderline cases in large-scale datasets.

Sofman et al. [44] developed *Anytime novelty detection algorithm*, a kernel-based algorithm designed to handle noisy, redundant, and high-dimensional feature spaces, commonly encountered in robotics. The approach transforms the challenge of environmental change detection into a location-specific novelty detection problem and uses Multiple Discriminant Analysis (MDA) instead of PCA to enhance robustness in high-dimensional spaces. It works as a variant of the NORMA (an online kernelized SVM) algorithm and is optimized using gradient descent by assuming new queries are novel until proven otherwise. It utilizes an anytime framework that ensures efficient computation with a fixed buffer size while reordering stored examples for faster adaptation, ensuring bounded computation time and effective anytime novelty prediction, even in complex environments. However, the performance depends on the accuracy of novelty assumptions and may generate higher false-positive rates in noisy environments.

2.3. Feature-based anytime algorithms

Xu et al. [45] introduced *Greedy Miser*, which is an anytime algorithm that minimizes test-time CPU usage by incorporating feature extraction costs into gradient boosting. It is an extension of stage-wise regression and achieves a balance between accuracy and computational efficiency by minimizing costly features without sacrificing prediction quality. It integrates cost-efficiency into the training and selection of weak classifiers, yielding a simple yet effective model. It can produce valid classification results with varying test-time budgets and progressively better predictions by incrementally adding features based on cost-benefit; it can stop early and still produce a valid output. The greedy miser algorithm for different values of $\lambda \in \{0, 1/4, 1/2, 1, 2, 4\}$ as feature-cost trade-off parameter (number of features in each cost group), yields an accuracy of more than 80% at the test-time cost of 25 s over the state-of-the-art approach Early-Exits. The lower the value of λ, the better the accuracy, but at higher costs. However, performance is highly sensitive to the accuracy of cost estimates, as the algorithm may discard high-cost but highly informative features when working under strict budgets, potentially leading to suboptimal accuracy.

Xu et al. [46] introduced the Anytime Feature Representation (*AFR*) algorithm that explicitly addresses the trade-off between using expensive features and evaluation cost in the data representation rather than the classifier. AFR enables conventional classifiers to become test-time cost-sensitive anytime classifiers; however, it introduces additional computational overhead due to the integrated representation learning stage. It combines the advantages of anytime learning and large margin classifiers, tackling the classification problem with a novel approach to budgeted learning. The algorithm consists of two integrated parts: classification using SVMs and feature representation learning using Greedy Miser. The feature representation mapping transforms the input vector into a new representation, which is then classified by the SVM within cost budgets. The anytime setting is achieved by incrementally increasing the cost budgets until the cost constraint no longer affects the optimal solution. AFR achieves the highest test scores overall, due to the generalization capabilities of large-margin classifiers.

2.4. Ensemble-based anytime algorithms

Ensemble methods are machine learning algorithms that construct a set of classifiers (base classifiers) and make predictions for new data points by combining the results of base classifiers, typically through a weighted voting mechanism or iterative refinement.

Kary et al. [47] developed the *EnE (Ears and Eyes)*, which is an anytime ensemble learning approach for sub-dialogue topic spotting. A classifier is designed to categorize short fragments of a conversation into one of several predefined topics provided during training. It aims for anytime classification by being biased towards faster performance over classification accuracy, analyzing the impact of test conversation length on the topic classification accuracy. It achieves anytime behavior by progressively improving classification accuracy as sub-dialogue length increases. The classifier leverages existing technologies, such as the BOOSTEXTER classifier and IBM's WATSON ASR system. The proposed two-phase training (verbatim + noisy ASR transcripts) with a single classifier is tuned for variable dialogue lengths. It achieves a limited classification accuracy of 45% on increasing the subdialogue length to "FULL", which significantly outperforms the majority-class baseline accuracy of 13%. Grubb et al. [25] devised the *SpeedBoost*, which is an extension of functional gradient descent to learn multiple anytime predictors, which are hypotheses that can automatically trade computation time for predicting accuracy with additional predictors. It essentially selects a set of weak predictors from simpler candidate models in an anytime fashion, efficiently allocating computational resources to more challenging examples by targeting specific data subsets. However, performance heavily depends on the quality/diversity of

candidate predictors because it uses extra resources at the time of prediction to generate a fast approximate result that can be improved by including a larger number of candidate predictors.

Building on this, *SpeedMachines* is another anytime technique proposed by Grubb et al. [48] for learning structured prediction. It accounts for both structural elements and feature computation during training, which influence the test-time inference. This approach automatically integrates new learners into predictors that enhance performance while optimizing efficiency in both feature and inference computation times. It refines predictions by incrementally updating only uncertain regions in structured outputs. The goal is to minimize a risk function by predicting portions of the output locally, which requires careful cost model tuning. Hierarchical Inference Machines (HIM) approach achieves 80% pixel classification accuracy with an increase in inference time from 10^{-1} to 10^{0} over Speedy Inference Machines (SIM). Wang et al. [26] introduced two novel anytime online boosting algorithms: OTB (Online Transfer Boosting) and OMB (Online Multitask Boosting). These algorithms are designed to handle data samples that arrive sequentially from different domains in batches, leveraging the knowledge of instances from other domains to enhance learning performance. They are implemented in an online fashion, which makes them anytime by approximating the normalization factor to provide intermediate models at any stage. The framework supports flexible base learners, allowing any online learner to be adapted for transfer or multitask learning, unlike existing non-anytime methods that are limited to specific learners. However, it is sensitive to domain/task similarity, and complexity increases with many tasks.

2.5. Neural networks-based anytime algorithms

Deep Neural Networks (DNNs) have emerged as one of the most versatile machine learning techniques, achieving state-of-the-art accuracy across a wide range of applications. However, this high level of accuracy comes with a significant computational cost, particularly when applying DNNs to new examples. For many tasks, the computational demands of DNNs have increased rapidly. Moreover, high test-time cost prevents DNNs from deploying on resource-constrained platforms. To mitigate this issue, anytime neural networks have been introduced, offering a solution that significantly reduces the computation time required for processing test data.

Bolukbasi et al. [49] presented an anytime deep neural network approach which adaptively reduces evaluation time on new examples without loss of accuracy. Instead of redesigning or approximating existing networks, two novel schemes were introduced: (1) adaptive network evaluation and (2) adaptive network selection. These schemes allow many examples to be correctly classified using fewer layers or lighter networks, thereby reducing the computational time. It achieves accuracy by adaptively selecting the number of layers or the network to evaluate per example, enabling partial inference when interrupted. However, performance depends on the reliability of early-exit decisions and misclassification in early layers cannot be recovered without full network evaluation. Lee et al. [28] introduced an approach for anytime neural prediction via slicing, where multiple thin sub-networks of the same depth are trained for the anytime prediction of test data. It leverages multi-branch residual DNNs by progressively removing branches while keeping the original depth. Then, each sub-network is trained with an independent batch normalization layer to ensure stability and accuracy across different capacities. The approach starts by evaluating the smallest sub-networks to make quick predictions and progressively moves to larger sub-networks as more resources become available, enabling progressive refinement in accuracy. It is observed that the difference between errors of shallow and thin networks increases as the required FLOPs decrease. However, smaller sub-networks suffer from higher error rates, especially under extreme FLOP reduction.

Huang et al. [29] presented a Multi-scale dense convolutional network *(MSDNet)*, which is a resource-efficient image classification method that optimizes CPU usage at test time through multi-branch network parameters. However, dense multi-branch structure increases training complexity and memory. It is based on two design principles: (1) generating and maintaining coarse-level features throughout the network and (2) interconnecting the layers with dense connectivity. These design principles allow the intermediate classifiers at a few layers to make early predictions at multiple depths without interference, improving accuracy incrementally as more FLOPs are spent.

Hu et al. [27] presented another approach, discussing the anytime predictions in DNNs, where an anytime predictor produces a sequence of more expensive and accurate predictions. In this approach, feature transformations are used to generate a sequence of intermediate features, which are then used for auxiliary predictions by using a prediction layer with parameters. Furthermore, it generates accurate anytime predictions increasingly with a loss weighting scheme to intermediate layers of the existing feed-forward networks without degrading the final performance. However, it requires a careful loss weighting scheme for intermediate predictions to avoid degrading final accuracy.

2.6. Probability estimation-based anytime algorithms

Naive Bayes classifiers are a group of classification algorithms based on Bayes' Theorem. Rather than being a single algorithm, it represents a family of algorithms that all share a common principle: the assumption that each pair of features is independent of the other.

Liu et al. [50] introduced an anytime algorithm for Bayesian network evaluation that modulates the granularity of state-space representations to balance accuracy and computational efficiency. It progressively refines the state space of variables and produces increasingly accurate approximations over time. The approach leverages the relationship between state-space granularity and computational demands, allowing for adaptive precision adjustments based on the needs of each problem instance. However, the REMB scoring function used to assess the quality of abstractions is computationally expensive for highly connected networks. Jitnah et al. [51] developed *treeNet*, which evaluates belief networks (BNs) using a two-step process: first, transforming the BN into a tree structure, with the query node at the root; second, performing anytime inference through treeNet search. Upon incorporating new evidence, the posterior probability of the query node is recalculated using a modified polytree message-passing algorithm through best-first search as more steps are processed, enabling real-time, incremental updates. However, initially designed for polytrees, requiring adaptation for general Bayesian networks.

Hulten et al. [52] proposed a scaling-up method for any induction algorithm based on discrete search, allowing the running time to become independent of database size while maintaining decision quality similar to that achieved with infinite data. The method works within predefined memory limits and requires only sequential data access, producing anytime results suitable for batch processing, streaming, time-changing, and active-learning applications. In the context of learning Bayesian networks, it significantly accelerates the learning process, achieving mining speeds of millions of examples per minute without compromising predictive performance. The framework is versatile, supporting various search types – including greedy, hill-climbing, and genetic algorithms – and enabling algorithms to function incrementally, within memory constraints, and adapt to changing data. However, performance depends on model complexity, and it is not inherently optimized for highly dynamic network structures.

Webb et al. [31] introduced the *Anytime AODE* algorithm, which is an extension of the AODE (Average of One-Dependence Estimators) algorithm, designed to provide conditional probability estimates for each class, rather than simply selecting a single class label. The goal of *AAODE* is to develop an algorithm similar to Naive Bayes (NB). This approach enables the algorithm to refine its initial class probability estimates incrementally through additional computation (by adding more SPODEs), up to a specified computational budget for

improved classification. The algorithm is inspired by the notion of n-dependence estimators and utilizes the Super-Parent One-Dependence Estimator(SPODE) to compute probability estimates. However, the performance gain diminishes with very large numbers of sub-models.

Yang et al. [53] devised *AAPE* (Anytime Averaged Probabilistic Estimators), which is an anytime classification algorithm, designed to adapt to varying computational resources in online applications. It incrementally improves classification accuracy by ensembling Bayesian probabilistic estimators, beginning with Naive Bayes (0-dependence) and adding Superparent-One-Dependence Estimators (SPODEs) as time allows. At each interruption point, AAPE provides averaged probability estimates, supporting robust classifications and incremental learning. However, it may require many SPODEs to match state-of-the-art accuracy.

Hui et al. [54] developed Anytime Averaged Probabilistic with Weight Estimator (*AAPWE*), which is an enhancement of the AAPE algorithm designed for online classification. Unlike traditional algorithms that require significant computational resources and time, AAPWE allows for immediate classifications by providing the best predicted class label based on averaged probabilities estimates. It incorporates weights for superparent attributes to improve classification effectiveness. However, effectiveness totally depends on the accurate weighting of superparent attributes.

Hui et al. [32] introduced Scheduling Anytime Average aged Probabilistic Estimators (*SAAPE*), a novel anytime classification framework that extends the AAPE algorithm. It is designed to classify a pool of instances, delivering accurate results even when interrupted, while optimizing collective classification performance. The framework uses seven scheduling schemes, such as First-Come-First-Served (FCFS), Round Robin, Minimum-margin instance first (MMIF), Controversial instance first, and Hybrid, to allocate computational resources efficiently to multiple instances, allowing for efficient and effective anytime classification. It handles multiple instances simultaneously, assigning priority values to each instance and allocating resources to the highest-priority instance. However, scheduling overhead may offset gains in small datasets.

2.7. Nearest neighbors-based anytime algorithms

k-Nearest Neighbors (k-NN) is a supervised classification algorithm that assigns a class label to test data objects based on their feature similarity with k objects that are closest to the test object. When applied to data stored on hierarchical data structures like R-trees, k-NN works as an efficient branch-and-bound traversal algorithm, optimizing the search process by systematically pruning irrelevant branches, thereby reducing the computational overhead while identifying the nearest neighbors [71].

Ueno et al. [16] designed a *SimpleRank* method to produce an instant classification result whose accuracy progressively improves with an increase in time allowance. This anytime classifier utilizes a heuristic-based method to sort the index of training data based on its contribution to classification and scans them sequentially until time runs out. This sorted index is then used to classify test data objects in an anytime manner. This approach is designed for streams; however, it can be applied to static datasets as well. However, performance may degrade if ranking does not align well with true relevance.

Xu et al. [30] introduced an *Anytime k-Nearest Neighbor* (k-NN) search algorithm utilizing MVP-trees. MVP-tree is a variant of R-trees, where the metric distance function determines domain-specific knowledge. In an MVP tree, median values are leveraged to create partitions, and the pivots of these partitions are stored as entries in the internal nodes, enhancing the efficiency of the search process. This method identifies k answers and improves the answer set monotonically to return an approximate solution on early termination. However, performance depends on the effectiveness of pivot selection and partitioning. The relative error of k-NN monotonically decreases from 0.25 and converges to 0 with an increase in cost, i.e. average number of distance calculations from 500 to 2500.

Aarti et al. developed *Any-k-NN* [38], which is an anytime hierarchical method for k-NN classification on data streams. This method uses a classification model *Any-NN-forest*, which is a collection of c *Any-NN-trees*, one tree for each of the c classes. *Any-NN-tree* stores a hierarchy of micro-clusters to summarize the training data objects, along with their class labels. This enables the algorithm to utilize micro-clusters at the lower levels of the tree to make decisions when the time allowance for inference is less. And, as the time allowance increases, the algorithm can traverse to the micro-clusters at the deeper levels of the tree to produce more accurate results. This enables Any-k-NN to handle very large data streams, incrementally updating its classification model, and effectively handle concept drift and class evolution. This method was originally proposed for handling data streams. However, it can be easily adapted to static datasets by disabling incremental updates. Any-MP-k-NN is the parallel variant for anytime k-NN classification of multi-port data streams over distributed memory architectures. It improves classification accuracy with an increase in the number of parallel streams (handled by separate computing nodes), achieves memory efficiency and handles very large, high-speed data streams effectively. The syncing process of Any-NN-forests across computing nodes involves encoding, communication, and aggregation steps, which could introduce dependencies.

Xi et al. [55] proposed another anytime algorithm that transforms the nearest neighbor classifier to provide immediate class predictions while allowing for increased accuracy with additional computational time. The framework prioritizes the examination of important exemplars—instances that are highly representative of a class – thereby optimizing resource allocation during classification. By employing a simple algorithm to establish a high-quality ordering of exemplars, the method can achieve substantial accuracy even when processing only a small fraction of the dataset. It achieves 90% accuracy on increasing the number of instances from 500 to 1500.

2.8. Active learning-based anytime algorithms

Donmez et al. [56] developed *IEThresh* (Interval Estimate Threshold), an anytime active learning framework for scenarios involving multiple noisy labelers of unknown accuracy. IEThresh uses interval estimation to dynamically assess each labeler's reliability by calculating a confidence interval around their estimated accuracy and then selecting the labeler(s) with the highest upper-bound confidence, and filters unreliable ones early. This approach balances exploration and exploitation by initially exploring multiple labelers to gauge accuracy, then increasingly relying on those identified as most reliable. IEThresh requires fewer queries to achieve a given level of accuracy with significantly reduced labeling effort. However, performance depends on having enough early queries to reliably estimate labeler quality.

Tomanek et al. [57] introduced cost-sensitive approaches to Active Learning (AL) to optimize annotation time and proposing three methods to incorporate annotation time into AL selection: a fixed cost budget, a linear benefit-cost rank, and a benefit-cost ratio to prioritize instances for annotation. The cost-sensitive methods proved especially advantageous in early rounds, suggesting the potential for even greater effectiveness in large annotation pools. However, it requires an accurate estimation of annotation times for each instance to perform well. It achieves an F-score of more than 85% on increasing the annotation time from 1000 to 6000 s over the FuSAL method.

Ramirez et al. [58] presented an Anytime Active Learning (*AAL*) approach that optimizes both annotation time and response rate by potentially interrupting annotators before they complete labeling instances. AAL aims to balance two competing objectives: minimizing annotation cost and maximizing the likelihood of receiving a label within a budget. The Static AAL strategy disregards the impact of sub-instance size k on label acquisition probability, while the Dynamic

AAL strategy models this probability to guide sub-instance selection. It is observed that Dynamic AAL adjusts sub-instance sizes based on utility and uncertainty, while Static AAL uses fixed sizes. However, it requires careful modeling of sub-instance size effects, and the static variant ignores these effects and may be suboptimal.

Ramirez et al. [59] introduced another approach which improves annotation efficiency by allowing annotators to label examples based on partial inspection, for instance, reading the first 25 words of a document instead of the entire text. By optimizing the balance between annotation time and label accuracy through dynamic interruption policies, the approach reduces annotation costs without sacrificing classifier performance. However, overly interrupting may reduce label accuracy. On annotating the first 25 words, it achieves AUC 0.792 (17% error reduction) within 3600s vs. AUC 0.752 when annotating 100 words.

2.9. Anytime algorithms for vision applications

Karayev et al. [19] devised a method for timely multi-class object detection in images to achieve optimal performance at any point between a specified start time and deadline. The approach uses a dynamic, closed-loop policy to infer image contents and decide which detector or classifier to deploy next. The system treats detectors and classifiers as black boxes, learning from execution traces through reinforcement learning. A new timeliness performance measure is introduced, enabling the policy to dynamically select actions that maximize recognition performance under time constraints. The system continuously updates its belief model based on observations, influencing the selection of subsequent actions, and aims to deliver the best possible recognition result if interrupted at any point between the setup time and the deadline. However, the approach depends on high-quality execution traces for effective policy learning.

Karayev et al. [60] developed another approach to optimize anytime performance in visual architectures by learning dynamic feature selection policies. It sequentially computes features and classifies at any stage, improving prediction quality with a budget. Decisions are made during test time based on observed data and intermediate results, with the approach leveraging a Markov Decision Process (MDP) and reinforcement learning. It aims to learn multiple classifiers for different clusters of feature sets, ensuring robustness to varying subsets; however, this increases the training complexity. Dynamic policy outperforms static baselines across budgets up to 60, improving the specificity of predictions at leaf nodes.

Liu et al. [61] presented a dynamic hierarchical model for anytime scene labeling. It allows for flexible trade-offs between efficiency and accuracy in pixel-level prediction, improving accuracy with increased budget. The approach incorporates feature computation and model inference costs, optimizing performance for any given test-time budget by learning a sequence of image-adaptive hierarchical models. The goal is to find an optimal selection policy that generates a sequence of hierarchical models with good performance at all possible test-time cost budgets. However, policy learning via MDP may be computationally expensive for very large datasets. It achieves 90% accuracy improvement over state-of-the-art scene parsing baselines on three semantic segmentation datasets with increasing test-time budgets.

Frohlich et al. [62] presented Iterative Context Forests *(ICF)*, a novel approach for contextual semantic segmentation using a tree-based framework that integrates local information with contextual knowledge. Designed for anytime scenarios, ICF allows for an interruption during the labeling process, enabling the immediate use of contextual cues after the first iteration. The method utilizes Random Decision Forests (RDF) to incorporate context directly during training without relying on Conditional Random Fields (CRF). However, it may require many iterations for maximum accuracy in highly complex scenes. It achieves 89.55% accuracy over the CRF baseline with an additional 1.74s computation per image.

2.10. Application-based other anytime algorithms for classification

Mark et al. [63] introduced an anytime, information-theoretic connectionist network which represents interactions between the predicting attributes and the classification attributes for feature selection, which incrementally improves feature subset quality over time. It is interruptible, providing a partial set of relevant features at any stage, with result quality measured by fuzzy information gain—a metric that aligns with user-perceived model quality. However, the approach is dependent on the effectiveness of fuzzy information gain as a quality metric, which may not capture all domain-specific relevance aspects. It reduces the tree size from 9 to 5 with minimal error increase (0.0% → 2.0%) using the C4.5 classifier.

Schlobach et al. [64] introduced an anytime classification algorithm based on approximate subsumption, a concept derived from ontology, and its performance is compared with classical subsumption using realistic benchmarks. This approach addresses the need for rapid query answers, with the quality of results improving as more time is allocated. The algorithm approximates the ontology O to answer the query rather than approximating the query Q itself, involving an approximation of terminological subsumption. This is achieved by interpreting the ontology in a non-standard way, using lower and upper approximations of an interpretation. The MORE strategy achieves the highest gains of more than 20% as iterations increased from 60 to 85; however, approximation accuracy depends heavily on ontology structure and concept frequency, potentially limiting applicability to sparse ontologies.

Bartok et al. [65] developed a novel anytime algorithm designed to achieve near-optimal regret in finite stochastic partial monitoring problems. The algorithm adapts dynamically to different difficulty levels within the opponent's strategy space, achieving minimax regret within logarithmic factors for both easy and hard instances while ensuring logarithmic individual regret in easy cases. By tailoring to stochastic game settings, where opponent outcomes are generated independently and identically distributed, the approach minimizes cumulative loss relative to the optimal action's expected loss. It achieves a minimax regret of 0; however, it requires stochastic opponent strategies and may degrade in adversarial settings.

Viet et al. [66] introduced an efficient method for speeding up the anytime time series classification by using motifs. Motifs, being much shorter subsequences of the time series, allow faster ordering. This reduces the computational cost of ordering instances for classification, which is typically high when using distance measures like Dynamic Time Warping (DTW). The process involves extracting motifs, ordering them using the SimpleRank method [16], and rearranging the training set accordingly. However, Motif-based ordering may reduce accuracy slightly compared to full DTW ordering.

3. Anytime clustering

Clustering is the task of assigning unlabeled objects into groups called clusters such that the similarity of objects within a group is maximized and the similarity of objects between different groups is minimized. Many clustering methods suffer from scalability issues on large datasets and do not support user involvement during run time. They include algorithms such as DBSCAN [72], S-means [73], etc. To address these issues, anytime clustering algorithms are proposed. An anytime clustering algorithm works by balancing execution time with the quality of the clustering results [34]. Fig. 8 summarizes the anytime algorithms proposed for clustering of large datasets. Table 3 highlights the anytime clustering algorithms along with their corresponding dataset characteristics and evaluation outcomes. These evaluations highlight the versatility of anytime algorithms across various data analytics tasks and demonstrate their ability to produce useful intermediate results with constrained resources. Our survey primarily focuses on identifying the presence or absence of parallel implementations across different tasks. Also, the discussion of parallel algorithms is qualitative and based on reported capabilities in the original papers, not on independently verified benchmarks.

Table 3
Datasets used for evaluation of anytime clustering, FI mining and recommendation systems that achieve the highest performance.

Algorithm	Dataset characteristics	Results
AnyDBC [35]	Name: Mallat, Size: 10 000	NMI Score : 0.824
AnyDBC-MC [74]	Name: PAMAP2, Size: 974 479	NMI Score : 1
A-DBSCAN [33]	Name: COIL20, Size: 10 000	NMI Score : 0.908
A-DBSCAN-XS [34]	Name: DS1 (fiber), Size: 1500	NMI Score : 0.988
AnyOPTICS [36]	Name: Lankersim, Size: 20k–99k	NMI Score : 1
SPARROW-SNN [75]	Name: Sequoia, Size: 62 556	Accuracy: 93%
Sakai et al. [76]	Name: PAMAP2, Size: 974 479	NMI Score : 1
COBRAS [77]	Name: Iris, Size: 150	Adjusted Rand Index : 1.0
Multi-User Static FI [14]	Name: Synthetic, Size : 20 000	Accuracy: >95%
FPOF [20]	Name: Chess, Size : 3196	Accuracy: 100%
ALPINE [78]	Name: Mushroom, Size : 8124	Time: <50 s
Arai et al. [67]	Name : IMDB, Size :26 784	Confidence: up to 95%
Ben-Shimon [68]	Name: MovieLens, Size: 1M	AUC : 80.34
Ben-Shimon et al. [69]	Name: MovieLens, Size: 1M	Precision: 0.5
KARPET [70]	Name: DBLP, Size:5M	Runtime : 2–5 ms

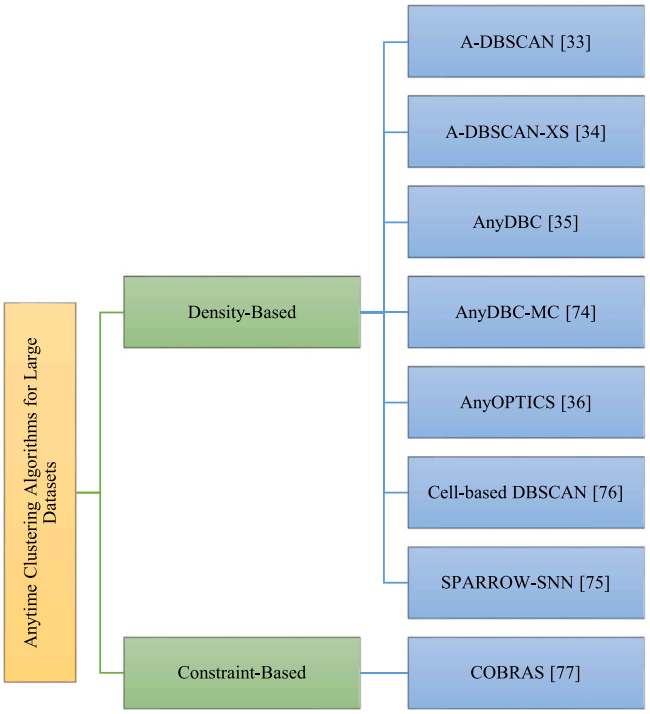

Fig. 8. Categorization of anytime clustering algorithms for large data.

3.1. Density-based anytime algorithms

Mai et al. [33] introduced *A-DBSCAN*, an anytime approach to the density-based clustering algorithm DBSCAN [72]. It leverages a sequence of lower-bounding functions (LBs) as a distance measure to generate multiple approximations of the final clustering result and significantly accelerate the computation of true density-based clusters. However, performance gains depend on the choice of lower-bounding functions and suboptimal LB sequences can slow convergence or reduce early-stage accuracy.

Mai et al. [34] developed *A-DBSCAN-XS*, an extended version of A-DBSCAN, designed to improve its performance by reducing distance calculations required at each level, making A-DBSCAN-XS more efficient and faster than A-DBSCAN. A-DBSCAN-XS builds upon the anytime scheme of A-DBSCAN and incorporates the μ-range query scheme of the extended Xseedlist data structure and updates only a subset of edges related to core objects at each level instead of all edges, reducing computational overhead. However, improvement in performance relies on the efficiency of the μ-range query scheme and subset edge updates; it may yield slightly slower refinement if the subset selection misses important edges early on. A-DBSCAN-XS is a single-threaded method which achieves a significant speedup compared to traditional DBSCAN and its variants. For example, on the dataset DS1, A-DBSCAN-XS achieves an NMI score of 0.842 at level 3 with a runtime of only 2.62 s, which is 112 times faster than DBSCAN (293.6 s). When it comes to the end, A-DBSCAN-XS requires only 5.3 s for DS1, which is 55 times faster than DBSCAN. For very large datasets, A-DBSCAN-XS demonstrates remarkable scalability. On a dataset with 20,000 fibers, A-DBSCAN-XS requires 1,284 s, which is 17 times faster than DBSCAN (21,292.4 s).

Mai et al. [35] presented *AnyDBC*, an anytime approach to the DBSCAN clustering algorithm which actively learns from data by iteratively analyzing the current cluster structure formation through some range queries. Unlike traditional methods that perform range queries on all objects, it refines the clusters at each iteration by selecting the most promising objects, thereby reducing both the number of range queries and label propagation time. However, it requires effective selection of promising objects, and performance can drop if the selection heuristic is poor.

Mai et al. [74] developed *AnyDBC-MC*, a parallel extension of AnyDBC, designed for shared memory architectures and enabling scalable parallel processing. It also improves the performance of AnyDBC by parallelizing the processing of queries in blocks and efficiently merging the results into the current cluster structure, allowing for more rapid clustering operations. As a result, AnyDBC-MC achieves orders of magnitude speedup even on a single thread compared to its sequential AnyDBC and demonstrates excellent scalability across multiple threads, maintaining high performance in multi-core environments. For the PAMAP2 dataset, AnyDBC take 346.9 s, while AnyDBC-MC (t = 1) takes 363.3 s. AnyDBC-MC is also orders of magnitude faster than other parallel algorithms like PDSDBSCAN and HPDBSCAN. For instance, AnyDBC-MC is up to 335.9 times faster than PDSDBSCAN on the Gas Sensor dataset with 16 threads. AnyDBC-MC exhibits near-linear scalability, achieving a 14.5× speedup on the Corpus dataset with 16 threads, with scalability further enhanced by larger block sizes (α, β) that increase per-thread workload and reduce synchronization overhead. In AnyDBC-MC, steps like merging (Step 4) and graph updates (Step 8) may suffer from load imbalance when the number of merge pairs is fewer than the number of threads, and scalability can be slightly reduced by NUMA effects.

Mai et al. [36] proposed *Any-OPTICS* for anytime OPTICS clustering, designed to improve performance in large-scale applications by reducing expensive distance computations. This is achieved by generating intermediate results and refining them continuously. Any-OPTICS works on multiple levels and uses the distance function to generate the reachability plot, which represents the ordering of objects. It essentially produces multiple reachability plots of various approximations

by employing a sequence of lower-bounding (LB) distances of the true distance function. However, performance depends on the choice of lower-bounding distances. *Any-OPTICS-XS* [36] is an extended version of Any-OPTICS, offering greater efficiency when handling highly expensive distance calculations. It relies on the monotonicity property of the reachability plots and a sequence of LBs, which helps to reduce the total number of distance calculations and enhance performance.

Sakai et al. [76] introduced *Anytime Cell-Based DBSCAN*, which is another anytime DBSCAN approach that enhances DBSCAN by dividing the dataset into smaller cells, which are then randomly selected and connected to compute clustering results rapidly. The algorithm iteratively refines these connections as more cells are processed to improve clustering accuracy and deliver precise results. However, clustering quality in early stages may depend heavily on the random selection of cells, potentially leading to less representative clusters until more iterations refine the connections.

Folino et al. [75] introduced the *SPARROW-SNN* approach, which is a distributed, biologically inspired clustering algorithm designed for peer-to-peer networks with a small-world topology. It combines an adaptive, flocking-based approach with a shared nearest-neighbor (SNN) clustering method to discover clusters independently across peers. By identifying dense regions in data and forming clusters around core points where local neighborhood density exceeds a specified threshold, it can detect clusters of varying shapes and sizes. Additionally, this decentralized, asynchronous approach supports incremental clustering and adapts well to large, distributed datasets. Essentially, this algorithm presents a trade-off between the number of peers used for the clustering and the accuracy of the clustering results. SPARROW-SNN achieves a significant reduction in execution time by processing a fraction of the dataset while scaling efficiently with more peers. By visiting only 5% of the data, accuracy drops only from 88% to 81%; for 10% of the visits, from 99% to 94%. This shows near-linear scalability in distributed settings since increasing peers reduces per-peer workload with minimal loss in clustering quality. The algorithm's design minimizes global dependencies by using local, asynchronous communication between peers and decentralized clustering, where peers work independently and they only merge via neighbor updates. These are the characteristics that generally enhance parallelism. However, communication costs may increase with peers.

3.2. Constraint-based anytime algorithm

Van et al. [77] presented the *COBRAS*(Constraint-Based Repeated Aggregation and Splitting) algorithm, an interactive constraint-based clustering approach. Being query-efficient and time-efficient, it combines user feedback with pairwise queries to create effective clusters. COBRAS works by grouping data into super-instances (small, locally coherent groups) that are assumed to belong to the same cluster. It iteratively refines the super-instances based on user feedback, enabling it to produce more granular clustering that improves over time, which makes it particularly suited for interactive and semi-supervised clustering tasks. However, effectiveness depends on the quality and representativeness of user-provided pairwise constraints, and performance can degrade if user feedback is noisy or biased.

4. Anytime frequent itemset mining

Anytime frequent itemset (FI) mining algorithms produce an approximate set of FIs whose approximation can be progressively improved as more processing time is available. There are three anytime FI mining algorithms for static data, as summarized in Fig. 9. The existing anytime methods have already exploited the use of statistical bounds and a sampling-based approach for quick and approximate results that improve over time. So, there is no future scope to improve, and thus leading to scarcity in FI mining. This scarcity can be primarily attributed to several factors, such as the computational cost, combinatorial complexity and the difficulty of generating meaningful, valid intermediate results without exhaustive computation. FI mining is inherently expensive and complex (exponential in worst-case) due to the combinatorial explosion of itemsets. The search space for itemsets is exponential in the number of items in candidate generation algorithms, due to which the number of FIs can grow combinatorially, especially when the minimum support threshold is low. Unlike classification (where early patterns support meaningful end results), partial FIs cannot reduce the complexity of space exploration.

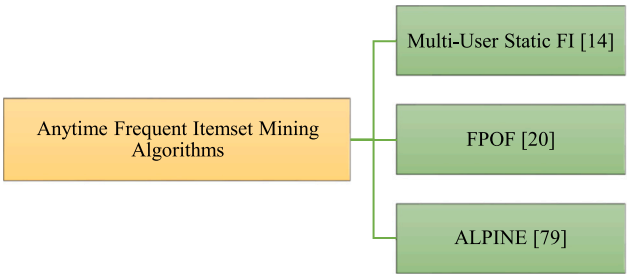

Fig. 9. Anytime frequent itemset mining algorithms.

Zhang et al. [14] developed an anytime static approach for mining large databases shared by multiple users. This technique identifies frequent itemsets (FI) in the database, gradually enhancing the quality of results as computation time increases. It provides progressively refined FIs, where early approximate mining outcomes can be accessed in an anytime manner. To support multiple-user inquiries, the approach leverages sampling and incremental mining, ensuring that intermediate results can be utilized effectively across users. This approach has been successfully applied to stock market data, utilizing the intermediate results generated by the anytime mining process. The method outperforms Apriori, with a consistent running time below 1500 s, even as incremental steps increase to 4. However, the accuracy of intermediate results depends on the sampling quality, and incremental updates may not capture sudden or highly dynamic changes in data distribution.

Giacometti et al. [20] introduced *FPOF*, an anytime approach for calculating the Frequent Pattern Outlier Factor, which is a metric used to detect anomalies in datasets. The method utilizes pattern sampling rather than exhaustive pattern mining, allowing the algorithm to produce approximations of FPOF with a guaranteed maximum error, based on statistical bounds like Bennett's inequality. This makes it suitable for large datasets, where exhaustive mining would be impractical. On a 500 K-transactional dataset, it achieved 90% accuracy within 10 s. FPOF is interruptible and resource-adaptive but sensitive to sampling bias, which can affect outlier detection accuracy.

Hu et al. [78] proposed ALPINE, a progressive anytime frequent itemset (FI) mining algorithm that provides definite guarantees during mining. Unlike traditional algorithms that only return complete results at the end, ALPINE incrementally enumerates FIs in a progressive order, ensuring that each discovered itemset is correct and final, with no need for retraction or reordering later. This is achieved by partitioning the search space into itemset intervals indexed by support and systematically exploring them in increasing frequency order, making the mining process both monotonic and complete over time. The algorithm ensures that at any interruption point, the set of itemsets discovered so far is definitive and requires no revision, ensuring reliable intermediate outputs and guaranteeing that they will be part of the final result. Although ALPINE guarantees correctness at all times, the overhead of maintaining support-indexed itemset intervals can reduce efficiency when handling extremely large, dense, or highly frequent datasets. This may lead to higher memory usage and slower performance compared to specialized non-anytime algorithms like FP-growth.

5. Anytime algorithms for recommendation systems

Platforms like Amazon, Netflix, and Spotify use recommender systems (RS) to personalize content or product suggestions for users. Anytime Algorithms can quickly provide recommendations based on a limited user profile or session data. It refines recommendations as more interaction data (clicks, views, ratings) is gathered in real-time.

Arai et al. [67] introduced an anytime framework for top-k queries on multi-attribute exact and fuzzy datasets, enhancing algorithms like TA (Threshold Algorithm) and TA-Sorted. The framework provides probabilistic quality measures (confidence that the current top-k is correct, precision as a lower bound on overlap with the true top-k, and score distance as an upper bound on score difference) at any execution point, allowing early termination with quantified guarantees. The approach uses data distribution models (e.g., histograms or probability density functions) to estimate unseen tuple scores probabilistically, ensuring monotonic improvement for TA (and expected monotonicity for TA-Sorted). The framework reduces runtime by allowing early termination with guarantees, with extensions for multidimensional histograms to handle attribute correlations. It ensures flexibility in trading off computation time for result quality, making these algorithms suitable for resource-constrained environments where early, usable results are preferred over waiting for full convergence. However, the method is limited by its reliance on accurate pre-computed or on-demand data distribution models, which may be unavailable or inaccurate for hidden or dynamic data sources, leading to unreliable guarantees.

Ben-Shimon [68] introduces anytime algorithms for top-N item–item collaborative filtering recommenders, addressing the needs of RS providers who must balance computational costs with model quality for outsourced services. The neighborhood-based CF is integrated into an anytime framework, where models can be interrupted at any time, yielding valid (suboptimal) results that improve monotonically and converge to optimality. Two ordering methods for item-pair similarity computations are proposed: (1) LSH Tree, which uses locality-sensitive hashing to build a balanced binary tree of item signatures, prioritizing likely similar pairs via bottom-up traversal and (2) Most Popular First, which sorts items by decreasing popularity (consumption set size) and computes similarities sequentially. These outperform a baseline (Amazon's arbitrary-order approach) in time-quality trade-offs. However, the approach focuses only on item–item similarity with the Jaccard coefficient, limiting generality to other similarity measures or model-based recommenders and is also limited by their reliance on implicit feedback datasets.

Ben-Shimon et al. [69] proposed another anytime algorithm for balancing computational costs and recommendation quality in RS, particularly for recommendation service providers. The algorithm generates recommendations of increasing quality as more time becomes available. The popular item–item top-N collaborative filtering approach is integrated into an anytime framework by prioritizing the order of item-pair similarity computations, ensuring valid suboptimal models at any interruption point while converging to optimality with sufficient time. This can be achieved by including two size-based heuristics for ordering pairs: an upper bound estimate (favoring pairs with similar and larger consumption set sizes) and an expected Jaccard estimate (assuming uniform user consumption distributions for precomputation) to prioritize item-pair computations, with efficient implementations via grouping or precalculation. However, the approach is sensitive to highly skewed consumption distributions, though the methods scale well to large catalogs.

Yang et al. [70] proposed KARPET (Kernelization1 And Rapid Pruning-based Exploration for Tree patterns), an any-k algorithm for retrieving top-k tree pattern matches in labeled graphs (heterogeneous information networks, HINs). Unlike traditional top-k algorithms requiring a fixed k, KARPET is an anytime algorithm that quickly returns the highest-ranked tree pattern matches (based on edge/node weights) and progressively delivers subsequent lower-ranked matches, allowing termination at any point with valid results. It tackles the NP-complete subgraph isomorphism problem for acyclic query patterns by leveraging: (1) aggressive pruning of the search space using the Yannakakis algorithm for homomorphic patterns, (2) a dynamic programming-based bottom-up cost calculation followed by top-down guided search, and (3) filtering to ensure isomorphism (no node repetition). KARPET exploits label constraints to reduce the gap between homomorphism and isomorphism, achieving millisecond-level response times for top results on graphs with millions of nodes/edges. However, KARPET is designed for tree (acyclic) query patterns, limiting its applicability to cyclic patterns common in some graph applications.

6. Anytime algorithms for streaming data

Nowadays, the volume, velocity, and variety of data being generated have grown exponentially, particularly with the advent of Internet of Things (IoT) devices, real-time monitoring systems, and social media platforms. This has led to an increasing demand for streaming data analytics, where streaming data is characterized by the continuous arrival of a sequence of data objects at a fast and variable rate. They are dynamic in nature, with real-time data arrival, and require processing of data objects on the fly as they arrive rather than being stored and analyzed in batches. Streaming data presents unique challenges, such as the need for real-time processing, limited memory resources, and the ability to handle evolving data distributions (concept drift). Traditional batch-processing algorithms are unsuitable for these scenarios, as they require the entire dataset to be available upfront and are not designed to provide intermediate results or adapt to evolving data distributions.

Anytime streaming algorithms have emerged as a solution to these challenges. They are designed to handle varying inter-arrival rates of data in the streams. Essentially, they produce approximate but valid results when the stream speed is high and produce more accurate results when the stream speed is low. Most of them take advantage of hierarchical indexing structures in order to achieve the anytime features. The key characteristics of these algorithms are similar to anytime algorithms for large datasets; (i) Interruptibility, i.e. the ability to provide valid results at any point during execution, even if interrupted. In addition to interruptibility, these algorithms also have properties like (ii) Resource Adaptiveness, i.e., the ability to adjust to varying levels of computational resources (e.g., time, memory) and produce results that improve as more resources become available. (iii) Incremental Learning, i.e. the ability to update models incrementally as new data arrives, without requiring a complete reprocessing of the entire data and (iv) Scalability, i.e. the ability to handle high-speed data streams and large volumes of data efficiently, often through the use of hierarchical indexing structures or parallel processing techniques. They are applied across a wide range of data analytics tasks, including classification, clustering, frequent itemset mining and anomaly detection. Anytime streaming algorithms include — Anytime Nearest Neighbors [16,17], Anytime Bayesian classifiers [39,79,80], Anytime Setwise Classification [40], ClusTree [12], LiarTree [81], SubClustree [82], AnyFI [15,83], etc. Anytime streaming algorithms efficiently process high-speed data streams and can be widely used in applications that include — Financial Analytics (fraud detection and stock market analysis) [84], Cybersecurity (intrusion detection and malware classification) [40], Healthcare Monitoring (patient vital sign tracking and anomaly detection), IoT & Smart Cities for Traffic monitoring, Air Quality Assessment, Energy Consumption Optimization, Social Media Analytics (sentiment analysis and event detection in live streams), etc. A prominent real-time application is network traffic monitoring for anomaly detection. In such systems, massive and unpredictable volumes of traffic, evolving cyber threats, and limited processing resources on network edge devices (routers, firewalls) demand anytime streaming algorithms that can dynamically adapt to varying data rates, handle concept drift, and maintain real-time responsiveness without

compromising on detection accuracy. In such scenarios, anytime algorithms have emerged as a solution. Specifically, when the system is under high load or working under constrained hardware (like edge routers), anytime algorithms quickly generate a coarse-grained analysis to detect anomalies without stalling. As more resources or time become available, either because the traffic eases or the system reprioritizes tasks, the algorithm incrementally refines its earlier output, enabling more precise anomaly detection. This progressive refinement allows the system to remain responsive under pressure while still improving detection quality over time. Another application is fraud detection in credit card transactions [20]. These systems monitor financial transactions to detect suspicious or anomalous behavior, where delays could lead to financial loss. Anytime algorithms can detect suspicious activity early and revise it if additional data (e.g., user behavior history) is given.

Anytime streaming algorithm, such as ClusTree [12], is an anytime hierarchical clustering algorithm designed for evolving data streams. It maintains aggregated micro-clusters in a tree that is updated incrementally, allowing it to provide clustering results at anytime. Its design efficiently handles variable stream speeds, concept drift and limited memory, though it is tailored specifically for clustering In contrast, AnySC [85] is an anytime setwise classification algorithm that employs a hierarchical CProf-forest structure to classify test entities in data streams within any given processing time allowance. It incrementally refines classification results using a best-first traversal strategy and supports updates. AnyFI [83], on the other hand, focuses on anytime frequent itemset mining for transactional data streams, employing a Buffered Frequent Itemset Forest and tilted-time windows to produce intermediate results that improve with time, though with potentially high memory usage.

7. limitations of the survey

This survey presents a comprehensive review of anytime algorithms across various data analytics tasks, such as with a particular focus on classification, clustering, frequent itemset mining, etc. These tasks were chosen due to their prominence in anytime algorithm research, encompassing 56 articles published between 1996 and 2025. However, a few limitations remain:

- We have selected the algorithms based on Anytime properties, such as interruptibility, resource adaptiveness, incremental learning and scalability & focused on only core data analytics tasks.
- The literature on anytime algorithms for static data is not yet fully explored for all the tasks of data analytics; mainly, core tasks are focused on in this survey.
- The literature remains skewed towards classification tasks, while other domains, such as clustering, frequent itemset mining and especially parallel anytime algorithms, are relatively underexplored, mainly due to many computational challenges. This gap reflects both the limited existing work and the need for further exploration.
- The survey provides a primarily qualitative review and lacks empirical benchmarks comparing accuracy, speedup, or scalability across anytime algorithms.
- The issues related to industrial deployment, such as integration with existing systems and real-time constraints, are not deeply explored due to limited information in the literature.

8. Open issues and research opportunities in the design of anytime algorithms

Based on the survey conducted on anytime algorithms for data analytics, we have found the following aspects remain open and worthwhile to be further studied in the future.

- **Lack of Anytime variants for traditional algorithms**: While numerous algorithms exist in the literature for processing the large datasets, many of them do not have their respective anytime variants. Traditional algorithms such as Denclue [86], Shared Nearest Neighbors [87], RECOME [88], SLINK [89], CLINK and AverageLINK, etc, are designed to work within limited memory and computational time to achieve optimal results. However, they are not suitable for scenarios where real-time or progressive results are needed. Developing anytime variants of these traditional algorithms would significantly expand the applicability of the anytime algorithm to a broader range of data mining tasks.
- **Lack of Interruptible variants** Many algorithms reported in the literature are contract algorithms wherein the resource allocation (e.g., time, memory) must be known in advance. Examples include ID3-k & LSID3 [22], Anytime Cost-sensitive induction trees [23], Multi-User Static FI [14], etc. Although these algorithms are capable of handling interruptions and resource constraints within a fixed budget, they do not provide meaningful results at the intermediate stages. Moreover, they are computationally expensive and may require significant processing memory and power to generate the results. Hence, efficient interruptible variants of the above algorithms can be developed.
- **Need for Parallel and Scalable Anytime Implementations** Most algorithms in the literature are sequential in nature, which poses a memory and computational resource bottleneck while processing large datasets. Only a few algorithms actually support parallelism [34,74,75]. This opens up opportunities to develop parallel algorithms for various parallel architectures such as distributed memory, shared memory, hybrid and GP-GPU architectures. One can experiment on both data parallel and task parallel workflows for the designed algorithms while exploiting the above parallel hardware architectures. Parallelization can significantly benefit the anytime algorithms in terms of efficiency, scalability, and performance, especially while handling large datasets.
- **Potential Applications for Anytime Algorithms** There is always an open-ended opportunity to develop novel anytime algorithms for many use cases that involve quick decision-making in time-constrained or dynamic environments. For example, modern traffic systems aim to optimize vehicle routing and flow in real-time using data from sensors, GPS, and traffic cameras. Anytime algorithms help in generating quick route suggestions based on current traffic data for changing conditions like accidents, road closures, or congestion. If more time or data are given, the algorithm can refine the route further. Similarly, in the domain of cloud computing, anytime algorithms can dynamically allocate resources based on fluctuating/varying workloads, ensuring efficient performance even under varying demands of users. By leveraging their ability to provide immediate though approximate, valid solutions, anytime algorithms can enhance the performance or optimize the algorithm that may not be capable of responding swiftly to changing conditions. Another application is Object recognition [19], which involves identifying and classifying objects within images or videos. Anytime algorithms provide early but approximate object detection, which helps avoid delays in decision-making, especially in time-critical systems like self-driving cars.

9. Conclusion

This survey provides a comprehensive review of anytime algorithms for data analytics, emphasizing their role in addressing the demands of large-scale, time-sensitive, and resource-constrained environments. We systematically categorized and reviewed, in a detailed manner, a number of representative state-of-the-art anytime algorithms across

Table 4
Summary of anytime classification algorithms for large datasets. Acronyms used **IL**: Interruptiblity; **RA**: Resource Adaptiveness; **SI**: Supports Incremental Learning; **IP**: Existence of Parallel Version.

Algorithm	Year	Category	IL	RA	SI	IP
ID3-k & LSID3 [22]	2004	Induction-trees based		✓		
IIDT [37]	2005	Induction-trees based	✓	✓		
Esmeir et al. [41]	2007	Induction-trees based		✓		
ACT [23]	2007	Induction-trees based		✓		
Esmeir et al. [42]	2008	Induction-trees based		✓		
Esmeir et al. [43]	2011	Induction-trees based		✓		
DeCoste et al. [13]	2002	Kernel-based		✓		
Sofman et al. [44]	2011	Kernel-based		✓		
Greedy Miser [45]	2012	Feature-based	✓			
AFR [46]	2013	Feature-based			✓	
EnE [47]	2000	Ensemble-based		✓		
SpeedBoost [25]	2012	Ensemble-based		✓		
SpeedMachines [48]	2013	Ensemble-based			✓	
OTB & OMB [26]	2015	Ensemble-based			✓	
Anytime DNN [49]	2017	Neural Nets-based			✓	
Lee et al. [28]	2018	Neural Nets-based		✓		
MSDnet [29]	2018	Neural Nets-based	✓			
Hu et al. [27]	2019	Neural Nets-based		✓		
Anytime NB [50]	1996	Prob. Estimation-based		✓		
treeNet [51]	1997	Prob. Estimation-based	✓			
Hulten et al. [52]	2002	Prob. Estimation-based			✓	
AAODE [31]	2006	Prob. Estimation-based		✓		
AAPE [53]	2007	Prob. Estimation-based		✓		
AAPWE [54]	2008	Prob. Estimation-based		✓		
SAAPE [32]	2009	Prob. Estimation-based		✓		
SimpleRank [16]	2006	Nearest Neighbor-based		✓		
MVP-Trees [30]	2008	Nearest Neighbor-based	✓			
Any-*k*-NN [38]	2024	Nearest Neighbor-based	✓	✓	✓	✓
NN classifier [55]	2008	Nearest Neighbor-based		✓		
IEThresh [56]	2009	Active Learning-based		✓		
Tomanek et al. [57]	2010	Active Learning-based	✓			
AAL [58]	2013	Active Learning-based	✓			
Ramirez et al. [59]	2014	Active Learning-based	✓			
Karayev et al. [19]	2012	Vision Appl.-based	✓			
Anytime Scenes [60]	2014	Vision Appl.-based	✓			
Liu et al. [61]	2016	Vision Appl.-based	✓			
ICF [62]	2012	Vision Appl.-based	✓			
AFS [63]	2001	Application-based	✓			
Schlobach et al. [64]	2007	Application-based	✓			
APM [65]	2012	Application-based	✓			
Viet et al. [66]	2013	Application-based	✓			

Table 5
Summary of anytime algorithms for clustering, FI Mining and Recommendation Systems. Acronyms used **IL**: Interruptiblity; **RA**: Resource Adaptiveness; **SI**: Supports Incremental Learning and **IP**: Existence of Parallel Version.

Algorithm	Year	Category	IL	RA	SI	IP
A-DBSCAN [33]	2013	Density-based Clust.		✓		
A-DBSCAN-XS [34]	2015	Density-based Clust.	✓			✓
AnyDBC [35]	2016	Density-based Clust.		✓		
AnyDBC-MC [74]	2018	Density-based Clust.	✓			✓
AnyOPTICS [36]	2016	Density-based Clust.		✓		
SPARROW-SNN [75]	2009	Density-based Clust.	✓			✓
Sakai et al. [76]	2022	Density-based Clust.		✓		
COBRAS [77]	2018	Constraint-based Clust.		✓		
Multi-User Static FI [14]	2002	FI Mining		✓		
FPOF [20]	2016	FI Mining		✓		
ALPINE [78]	2017	FI Mining	✓		✓	
Aria et al. [67]	2009	Recommendation-based	✓			
Ben-Shimon [68]	2013	Recommendation-based	✓	✓		
Ben-Shimon et al. [69]	2016	Recommendation-based	✓	✓		
KARPET [70]	2018	Recommendation-based	✓		✓	

various data analytics tasks, designed to handle large datasets. These algorithms are categorized based on their foundational methodologies, covering a variety of approaches, including classification, clustering, and frequent itemset mining. This categorization highlights various algorithms employed for anytime data analytics and provides a structured overview of how different algorithms are suited to different data analytics tasks. Our literature survey analysis, based on 41 classification algorithms (1996–2025), 8 clustering algorithms (2009–2022), 3 frequent itemset mining algorithms (2002–2020) and 4 recommendation systems algorithms (2009–2018), shows that classification-focused anytime algorithms are the most mature, while clustering, frequent itemset mining and recommendation systems remain emerging areas with significant research potential. Despite the progress, key research challenges remain. We introduced metrics for performance evaluation of anytime algorithms and compared anytime algorithms with non-anytime alternatives, highlighting their advantages in time-sensitive scenarios.

Table 4 summarizes the anytime classification algorithms useful for large-scale data based on different machine learning models. They are compared based on a set of key factors, which helps in understanding their applicability and behavior in real-world scenarios. Table 5 lists the anytime clustering, frequent itemset mining and recommendation system algorithms proposed for large datasets. Overall Table 4, 5 summarizes all the algorithms presented in this paper while highlighting the specific features of each algorithm. They provide an overview of various anytime algorithms for data analytics in terms of their anytime capabilities, interruptibility, applications, and parallelism.

The survey also answers the questions outlined in Section 1.4. The key findings from the survey include the ability of the anytime algorithms to balance time constraints with solution quality, making them highly effective in large-scale, time-sensitive analytics tasks as seen in, for instance, hierarchical *k*-NN classifiers (*Any-k-NN*) (RQ1). The design of anytime algorithms is focused on four fundamental

principles for various tasks of data analytics: interruptibility, resource adaptiveness, solution quality, and scalability. To achieve these objectives, many anytime algorithms leverage incremental updates and hierarchical structures to handle large datasets (RQ2). It highlights the evolution of anytime algorithms from decision-tree and probabilistic models to more recent developments involving ensemble learning, deep neural networks, and distributed frameworks, showcasing their theoretical advancements over traditional methods (RQ3). This survey also highlights the significant challenges identified in anytime algorithms and future research opportunities (refer to Section 8) (RQ4). It also emphasises how parallelization techniques can enhance the scalability and efficiency of these algorithms (RQ5).

In conclusion, anytime algorithms represent a robust and evolving solution to the challenges of modern data analytics. Their capacity to trade off between computational effort and decision quality makes them not only theoretically appealing but also highly practical for a wide range of real-world applications.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used Grammarly in order to enhance grammar and clarity. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Funding sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, first ed., Addison-Wesley Longman Publishing Co., Inc., USA, 2006, p. 769.
[2] V. Kumar, R.S. Singh, M. Rambabu, Y. Dua, Deep learning for hyperspectral image classification: A survey, Comput. Sci. Rev. 53 (2024) 100658.
[3] D.V. Yevle, P.S. Mann, Artificial intelligence based classification for waste management: A survey based on taxonomy, classification & future direction, Comput. Sci. Rev. 56 (2025) 100723.
[4] S. El Khediri, W. Fakhet, T. Moulahi, R. Khan, A. Thaljaoui, A. Kachouri, Improved node localization using K-means clustering for Wireless Sensor Networks, Comput. Sci. Rev. 37 (2020) 100284.
[5] D. Adhikari, W. Jiang, J. Zhan, D.B. Rawat, A. Bhattarai, Recent advances in anomaly detection in Internet of Things: Status, challenges, and perspectives, Comput. Sci. Rev. 54 (2024) 100665.
[6] K.G. Al-Hashedi, P. Magalingam, Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019, Comput. Sci. Rev. 40 (2021) 100402.
[7] P.-N. Tan, M. Steinbach, V. Kumar, Finding spatio-temporal patterns in earth science data, Earth Sci. (2001) 1–12.
[8] V. Springel, S.D. White, A. Jenkins, C.S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J.A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, F. Pearce, Simulations of the formation, evolution and clustering of galaxies and quasars, Nature 435 (7042) (2005) 629–636.
[9] D.J. Hand, N.A. Heard, Finding groups in gene expression data, J. Biomed. Biotechnol. 2005 (2) (2005) 215–225.
[10] J. Pei, J. Han, B. Mortazavi-asl, H. Zhu, Mining access patterns efficiently from web logs, Curr. Issues New Appl. Knowl. Discov. Data Min. (2000) 396–407.
[11] S. Brin, R. Motwani, J.D. Ullman, S. Tsur, Dynamic itemset counting and implication rules for market basket data, SIGMOD Rec. (ACM Spec. Interes. Group Manag. Data) 26 (2) (1997) 255–264.
[12] P. Kranen, I. Assent, C. Baldauf, T. Seidl, The ClusTree: Indexing micro-clusters for anytime stream mining, Knowl. Inf. Syst. 29 (2) (2011) 249–272.
[13] D. DeCoste, Anytime interval-valued outputs for kernel machines: Fast support vector machine classification via distance geometry, in: Proc. of the 19th International Conference on Machine Learning, Vol. 9, 2002, pp. 99–106.
[14] S. Zhang, C. Zhang, Anytime mining for multiuser applications, IEEE Trans. Syst. Man Cybern. A:Syst. Hum. 32 (4) (2002) 515–521.
[15] P. Goyal, J.S. Challa, S. Shrivastava, N. Goyal, Anytime frequent itemset mining of transactional data streams, Big Data Res. 21 (2020) 100146.
[16] K. Ueno, A. Xi, E. Keogh, D.J. Lee, Anytime classification using the nearest neighbor algorithm with applications to stream mining, in: Proceedings - IEEE International Conference on Data Mining, 2006, pp. 623–632.
[17] C.I. Lemes, D.F. Silva, G.E. Batista, Adding diversity to rank examples in anytime nearest neighbor classification, in: Proc. of the 13th International Conference on Machine Learning and Applications, ICMLA, 2014, pp. 129–134.
[18] G. Nagy, A.R. Várkonyi-Kóczy, J. Tóth, An anytime voice controlled ambient assisted living system for motion disabled persons, in: Proc. of the International Symposium on Medical Measurements and Applications (MeMeA), IEEE, 2015, pp. 163–168.
[19] S. Karayev, T. Baumgartner, M. Fritz, T. Darrell, Timely object recognition, in: Proc. of the Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2012, pp. 890–898.
[20] A. Giacometti, A. Soulet, Anytime algorithm for frequent pattern outlier detection, Int. J. Data Sci. Anal. 2 (2016) 119–130.
[21] L.H.O. Rios, L. Chaimowicz, A survey and classification of a* based best-first heuristic search algorithms, in: Proc. of the Brazilian Symposium on Artificial Intelligence, Springer, 2010, pp. 253–262.
[22] S. Esmeir, S. Markovitch, Lookahead-based algorithms for anytime induction of decision trees, in: Proc. of the 21st International Conference on Machine Learning, 2004, p. 33.
[23] S. Esmeir, S. Markovitch, Anytime induction of cost-sensitive trees, in: Proc. of the Conference on Advances in Neural Information Processing Systems, 2007, pp. 1–8.
[24] K. Myers, M. Kearns, S. Singh, M.a. Walker, A boosting approach to topic spotting on subdialogues, in: Proc. of the 17th International Conference on Machine Learning, 2000, pp. 655–662.
[25] A. Grubb, J.A. Bagnell, SpeedBoost: Anytime prediction with uniform near-optimality, J. Mach. Learn. Res. 22 (2012) 458–466.
[26] B. Wang, J. Pineau, Online boosting algorithms for anytime transfer and multitask learning, in: Proc. of the AAAI Conference on Artificial Intelligence, Vol. 29, 2015, pp. 3038–3044.
[27] H. Hu, D. Dey, M. Hebert, J.A. Bagnell, Learning anytime predictions in neural networks via adaptive loss balancing, in: Proc. of the 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 3812–3821.
[28] H. Lee, J. Shin, Anytime neural prediction via slicing networks vertically, 2018, pp. 1–13, ArXiv arXiv:1807.0.
[29] G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, K. Weinberger, Multi-scale dense networks for resource efficient image classification, in: Proc. of the 6th International Conference on Learning Representations, 2018, pp. 1–14.
[30] W. Xu, D.P. Miranker, R. Mao, S. Ramakrishnan, Anytime K-nearest neighbor search for database applications, in: Proc. of the 24th International Conference on Data Engineering Workshop, 2008, pp. 426–435.
[31] G.I. Webb, J.R. Boughton, Y. Yang, Learning for anytime classification, in: Proc. of the AAAI Conference on Artificial Intelligence, 2006, pp. 1–6.
[32] B. Hui, Y. Yang, G.I. Webb, Anytime classification for a pool of instances, Mach. Learn. 77 (1) (2009) 61–102.
[33] S.T. Mai, X. He, J. Feng, C. Böhm, Efficient anytime density-based clustering, in: Proc. of the International Conference on Data Mining, SIAM, 2013, pp. 112–120.
[34] S.T. Mai, X. He, J. Feng, C. Plant, C. Böhm, Anytime density-based clustering of complex data, Knowl. Inf. Syst. 45 (2) (2015) 319–355.
[35] S.T. Mai, I. Assent, M. Storgaard, AnyDBC : An efficient anytime density-based clustering algorithm for very large complex datasets, in: Proc. of the 22nd SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1025–1034.
[36] S.T. Mai, I. Assent, A. Le, Anytime OPTICS: An efficient approach for hierarchical Density-Based clustering, in: Proc. of the 21st International Conference on Database Systems for Advanced Applications, Springer, 2016, pp. 164–179.
[37] S. Esmeir, S. Markovitch, Interruptible anytime algorithms for iterative improvement of decision trees, in: Proc of the 1st International Workshop on Utility-Based Data Mining, 2005, pp. 78–85.
[38] Aarti, J.S. Challa, H. Harsh, U. Darolia, M. Agarwal, R. Chaudhary, N. Goyal, P. Goyal, A hierarchical anytime k-NN classifier for large-scale high-speed data streams, in: Proc. of the 16th International Conference on Agents and Artificial Intelligence, 2024, pp. 276–287.
[39] T. Seidl, I. Assent, P. Kranen, R. Krieger, J. Herrmann, Indexing density models for incremental learning and anytime classification on data streams, in: Proc. of the 12th International Conference on Extending Database Technology, 2009, pp. 311–322.

[40] J.S. Challa, P. Goyal, V.M. Giri, D. Mantri, N. Goyal, AnySC: Anytime Set-wise classification of variable speed data streams, in: Proc. of the International Conference on Big Data, IEEE, 2019, pp. 967–974.

[41] S. Esmeir, S. Markovitch, Anytime learning of decision trees, J. Mach. Learn. Res. 8 (5) (2007) 891–933.

[42] S. Esmeir, S. Markovitch, Anytime induction of low-cost, low-error classifiers: A sampling-based approach, J. Artificial Intelligence Res. 33 (2008) 1–31.

[43] S. Esmeir, S. Markovitch, Anytime learning of anycost classifiers, Mach. Learn. 82 (3) (2011) 445–473.

[44] B. Sofman, B. Neuman, A. Stentz, J.A. Bagnell, Anytime online novelty and change detection for mobile robots, J. Field Robot. 28 (4) (2011) 589–618.

[45] Z. Xu, K. Weinberger, O. Chapelle, The greedy miser: Learning under test-time budgets, 2012, arXiv preprint arXiv:1206.6451.

[46] Z. Xu, M.J. Kusner, G. Huang, K.Q. Weinberger, Anytime representation learning, in: Proc. of the 30th International Conference on Machine Learning, Vol. 28, 2013, pp. 2113–2121.

[47] K.M. Kary, S. Singh, A boosting approach to topic spotting on subdialogues, in: Proc. of the 17th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, USA, 2000, pp. 1–8.

[48] A. Grubb, D. Munoz, J.A. Bagnell, M. Hebert, SpeedMachines: Anytime structured prediction, 2013, pp. 1–17, arXiv preprint arXiv:1312.0579.

[49] T. Bolukbasi, J. Wang, O. Dekel, V. Saligrama, Adaptive neural networks for efficient inference, in: Proc. of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 527–536.

[50] C.-L. Liu, M.P. Wellman, On state-space abstraction for anytime evaluation of Bayesian networks, ACM SIGART Bull. 7 (2) (1996) 50–57.

[51] N. Jitnah, A. Nicholson, Treenets: A framework for anytime evaluation of belief networks, in: Proc. of the International Joint Conference on Qualitative and Quantitative Practical Reasoning, Springer, 1997, pp. 350–364.

[52] G. Hulten, P. Domingos, Mining complex models from arbitrarily large databases in constant time, in: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 525–531.

[53] Y. Yang, G. Webb, K. Korb, K.M. Ting, Classifying under computational resource constraints: anytime classification using probabilistic estimators, Mach. Learn. 69 (2007) 35–53.

[54] B. Hui, Y. Wu, A new anytime classifier basing on AAPE: Anytime averaged probabilistic with weight estimator (AAPWE), in: Proc. of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE, 2008, pp. 1–5.

[55] X. Xi, K. Ueno, E. Keogh, D.-J. Lee, Converting non-parametric distance-based classification to anytime algorithms, Pattern Anal. Appl. 11 (2008) 321–336.

[56] P. Donmez, J.G. Carbonell, J. Schneider, Efficiently learning the accuracy of labeling sources for selective sampling, in: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 259–268.

[57] K. Tomanek, U. Hahn, A comparison of models for cost-sensitive active learning, in: Coling 2010: Posters, 2010, pp. 1247–1255.

[58] M.E. Ramirez-Loaiza, A. Culotta, M. Bilgic, Towards anytime active learning: Interrupting experts to reduce annotation costs, in: Proc. of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, 2013, pp. 87–94.

[59] M.E. Ramirez-Loaiza, A. Culotta, M. Bilgic, Anytime active learning, in: Proc. of the National Conference on Artificial Intelligence, Vol. 3, 2014, pp. 2048–2054.

[60] S. Karayev, M. Fritz, T. Darrell, Anytime recognition of objects and scenes, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 572–579.

[61] B. Liu, X. He, Learning dynamic hierarchical models for anytime scene labeling, in: Lecture Notes in Computer Science, in: LNCS, vol. 9910, 2016, pp. 650–666.

[62] B. Fröhlich, E. Rodner, J. Denzler, As time goes by—anytime semantic segmentation with iterative context forests, in: Proc. of the Joint 34th DAGM and 36th OAGM Symposium on Pattern Recognition, Springer, 2012, pp. 1–10.

[63] M. Last, A. Kandel, O. Maimon, E. Eberbach, Anytime algorithm for feature selection, in: Proc. of the 2nd International Conference on Rough Sets and Current Trends in Computing, RSCTC, Springer, 2001, pp. 532–539.

[64] S. Schlobach, E. Blaauw, M. El Kebir, A. Ten Teije, F. Van Harmelen, S. Bortoli, M. Hobbelman, K. Millian, Y. Ren, S. Stam, P. Thomassen, R. Van Het Schip, W. Van Willigem, Anytime classification by ontology approximation, in: CEUR Workshop Proceedings, Vol. 291, 2007.

[65] G. Bartók, N. Zolghadr, C. Szepesvári, An adaptive algorithm for finite stochastic partial monitoring, 2012, arXiv preprint arXiv:1206.6487.

[66] N.Q. Viet, D.T. Anh, Using motif information to improve anytime time series classification, in: Proc. of the IEEE International Conference on Soft Computing and Pattern Recognition (SoCPaR), 2013, pp. 1–6.

[67] B. Arai, G. Das, D. Gunopulos, N. Koudas, Anytime measures for top-k algorithms on exact and fuzzy data sets, VLDB J. 18 (2) (2009) 407–427.

[68] D. Ben-Shimon, Anytime algorithms for top-N recommenders, in: Proceedings of the 7th ACM Conference on Recommender Systems, 2013, pp. 463–466.

[69] D. Ben-Shimon, L. Rokach, G. Shani, B. Shapira, Anytime algorithms for recommendation service providers, ACM Trans. Intell. Syst. Technol. (TIST) 7 (3) (2016) 1–26.

[70] X. Yang, D. Ajwani, W. Gatterbauer, P.K. Nicholson, M. Riedewald, A. Sala, Any-k: Anytime top-k tree pattern retrieval in labeled graphs, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 489–498.

[71] G.R. Hjaltason, H. Samet, Distance browsing in spatial databases, ACM Trans. Database Syst. 24 (2) (1999) 265–318.

[72] F. Cao, M. Estert, W. Qian, A. Zhou, Density-based clustering over an evolving data stream with noise, in: Proc. of the 6th International Conference on Data Mining, SIAM, 2006, pp. 328–339.

[73] P. Li, Y.g. Ding, P.p. Yao, K.m. Xue, C.m. Li, Some methods for classification and analysis of multivariate observations, J. Mater. Eng. Perform. 25 (8) (2016) 3439–3447.

[74] S.T. Mai, I. Assent, J. Jacobsen, M.S. Dieu, Anytime parallel density-based clustering, Data Min. Knowl. Discov. 32 (4) (2018) 1121–1176.

[75] G. Folino, A. Forestiero, G. Spezzano, Distributed anytime clustering using biologically inspired systems, in: Proc. of the IEEE International Conference on Adaptive and Intelligent Systems, 2009, pp. 120–125.

[76] T. Sakai, K. Tamura, H. Kitakami, T. Takezawa, Anytime Cell-based DBSCAN algorithm that connects randomly selected cells and its performance evaluation, Int. J. Serv. Knowl. Manag. 6 (1) (2022).

[77] T. Van Craenendonck, S. Dumančić, E. Van Wolputte, H. Blockeel, COBRAS: interactive clustering with pairwise queries, in: Proc. of the 17th International Symposium on Advances in Intelligent Data Analysis, Springer, 2018, pp. 353–366.

[78] Q. Hu, T. Imielinski, Alpine: Progressive itemset mining with definite guarantees, in: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, 2017, pp. 63–71.

[79] P. Kranen, S. Günnemann, S. Fries, T. Seidl, MC-tree: Improving bayesian anytime classification, in: Proc. of the 22nd International Conference on Scientific and Statistical Database Management, Springer, 2010, pp. 252–269.

[80] P. Kranen, M. Hassani, T. Seidl, BT* - An advanced algorithm for anytime classification, Lecture Notes in Comput. Sci. 7338 LNCS (2012) 298–315.

[81] M. Hassani, P. Kranen, T. Seidl, Precise anytime clustering of noisy sensor data with logarithmic complexity, in: Proc. of the 5th International Workshop on Knowledge Discovery from Sensor Data, 2011, pp. 52–60.

[82] M. Hassani, P. Kranen, R. Saini, T. Seidl, Subspace anytime stream clustering, in: Proc. of the 26th International Conference on Scientific and Statistical Database Management, 2014, pp. 1–4.

[83] P. Goyal, J.S. Challa, S. Shrivastava, N. Goyal, AnyFI: An anytime frequent itemset mining algorithm for data streams, in: Proc. of the IEEE International Conference on Big Data, 2017, pp. 942–947.

[84] I. Assent, P. Kranen, C. Baldauf, T. Seidl, Detecting outliers on arbitrary data streams using anytime approaches, in: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 10–15.

[85] J.S. Challa, P. Goyal, A. Kokandakar, D. Mantri, P. Verma, S. Balasubramaniam, N. Goyal, Anytime clustering of data streams while handling noise and concept drift, J. Exp. Theor. Artif. Intell. 34 (3) (2022) 399–429.

[86] A. Hinneburg, H.H. Gabriel, DENCLUE 2.0: Fast clustering based on kernel density estimation, Lecture Notes in Comput. Sci. 4723 LNCS (2007) 70–80.

[87] S. Kumari, S. Maurya, P. Goyal, S.S. Balasubramaniam, N. Goyal, Scalable parallel algorithms for shared nearest neighbor clustering, in: Proc. of the 23rd International Conference on High Performance Computingv(HiPC), IEEE, 2017, pp. 72–81.

[88] Y.-a. Geng, Q. Li, R. Zheng, F. Zhuang, R. He, N. Xiong, RECOME: A new density-based clustering algorithm using relative KNN kernel density, Inform. Sci. 436 (2018) 13–30.

[89] Robin Sibson, A Density-Based algorithm for discovering clusters in large spatial databases with noise, Comput. J. 16 (1973) 30–34.