SKYLINE QUERY PROCESSING FOR LARGE-SCALE AND INCOMPLETE GRAPHS USING GRAPH CONVOLUTIONAL NETWORK (GCN)

BY

HASAN KHAIR BIN ADZMAN

A thesis submitted in fulfilment of the requirement for the degree of Master of Computer Science and Technology

Kulliyyah of Information and Communication Technology
International Islamic University Malaysia

AUGUST 2025

ABSTRACT

Skyline query processing is essential in multi-criteria decision-making, as it retrieves optimal results without requiring user-defined weights. Traditional skyline methods, however, face significant challenges when applied to large-scale and incomplete datasets. This study proposes a hybrid approach that integrates the ISkyline dominance graph technique with Graph Neural Networks (GNNs), specifically a Graph Convolutional Network (GCN) to improve skyline query performance under such conditions. The GCN component is utilized to predict skyline tuples in the presence of missing or incomplete data. The ISkyline algorithm serves as the foundation for identifying initial dominance relationships and labelling skyline points, enabling the GCN to learn Pareto-optimal patterns from partially incomplete data. Evaluation on both synthetic and real-world datasets demonstrates enhanced accuracy and efficiency when compared to established methods such as ISkyline, SIDS, and OIS. The proposed GNN + ISkyline framework improved classification accuracy by 72%, the F1-score by 71%, and the AUC-ROC by 49% compared to the standalone ISkyline algorithm when evaluated on the CoIL 2000 dataset. This work demonstrates the potential of creating a more efficient query processing, supporting applications in e-commerce, finance, and smart data systems, while aligning with the 9th Sustainable Development Goal on industry, innovation, and infrastructure.

خلاصة البحث

تعد معالجة استعلامات الأفق Skyline ضرورية في اتخاذ القرارات متعددة المعايير، حيث إنها تسترد النتائج المثلى دون الحاجة إلى أوزان محددة من قبل المستخدم. ومع ذلك، تواجه طرق معالجة الاستعلامات التقليدية تحديات كبيرة عند تطبيقها على مجموعات بيانات غير كاملة وواسعة النطاق. تقترح هذه الدراسة نهجًا هجينًا يدمج تقنية الرسم البياني المهيمنة ISkyline مع الشبكات العصبية للرسم البياني (GNNs)، وبشكل خاص شبكة الرسم البياني التلافيفية (GCN) لتحسين أداء استعلام الأفق في ظل هذه الظروف. يتم استخدام مكون GCN للتنبؤ بمجموعات معالجة استعلامات الأفق في حالة وجود بيانات مفقودة أو غير كاملة. تُستخدم خوارزمية ISkyline كأساس لتحديد علاقات الهيمنة الأولية ووضع علامات على نقاط الأفق، مما يتيح لـ GCN تعلم الأنماط المثالية لباريتو من البيانات غير المكتملة جزئيًا. يُظهر التقييم على كلٍّ من مجموعات البيانات الاصطناعية والواقعية دقةً وكفاءةً مُحسّنتين عند مقارنتهما بالطرق المعتمدة مثل ISkyline وأفق البيانات غير المكتملة القائم على الفرز SIDS والأفق غير المكتمل المحسَّن OIS. وقد حسّن هيكل الشبكات العصبية للرسم البياني مع ISkyline (GNN + ISkyline) المقترح دقة التصنيف بنسبة 72%، ودرجة F1 بنسبة 71%، والمساحة تحت منحنى خاصية الاستقبال والتشغيل AUC-ROC بنسبة 49%، مقارنةً بخوارزمية ISkyline المستقلة عند تقييمها على مجموعة بيانات CoIL 2000. يوضح هذا العمل إمكانية إنشاء معالجة استعلامات أكثر كفاءة، ودعم التطبيقات في التجارة الإلكترونية، والتمويل، وأنظمة البيانات الذكية، مع التوافق مع الهدف التاسع من أهداف التنمية المستدامة بشأن الصناعة والابتكار والبنية التحتية.

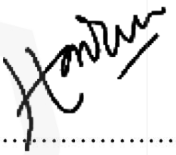
APPROVAL PAGE

I certify that I have supervised and read this study and that in my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Computer Science and Technology.

.....

Dr. Raini Binti Hassan

Supervisor

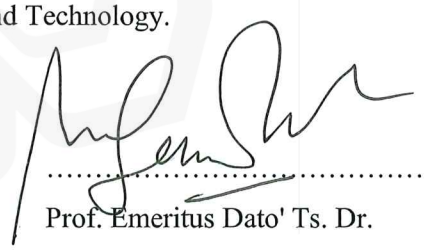


.....

Ts. Dr. Dini Oktarina Dwi Handayani

Co-Supervisor

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Computer Science and Technology.



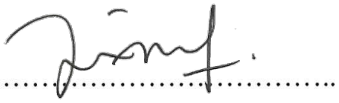
.....

Prof. Emeritus Dato' Ts. Dr. Tengku Mohd Bin Tengku Sembok

Internal Examiner

.....

Assoc. Prof. Dr. Adamu
Abubakar Ibrahim
Internal Examiner



.....

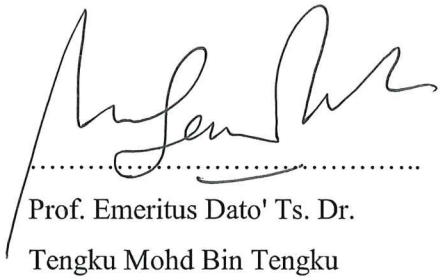
Assoc. Prof. Dr. Mohammad
Faidzul Nasrudin
External Examiner

This thesis was submitted to the Department of Computer Science and is accepted as a fulfilment of the requirement for the degree of Master of Computer Science and Technology.

.....

Dr. Azlin Binti Nordin
Head, Department of Computer Science

This thesis was submitted to the Kulliyyah of Information and Communication Technology and is accepted as a fulfilment of the requirement for the degree of Master of Computer Science and Technology.



Prof. Emeritus Dato' Ts. Dr.
Tengku Mohd Bin Tengku
Sembok Dean, Kulliyyah of
Information and Communication
Technology

DECLARATION

I hereby declare that this dissertation is the result of my own investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at IIUM or other institutions.

Hasan Khair bin Adzman

Signature: ..... Date: 19 August 2025

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

DECLARATION OF COPYRIGHT AND AFFIRMATION OF FAIR USE OF UNPUBLISHED RESEARCH

SKYLINE QUERY PROCESSING FOR LARGE-SCALE AND INCOMPLETE GRAPHS USING GRAPH CONVOLUTIONAL NETWORK (GCN)

I declare that the copyright holders of this dissertation are jointly owned by the student and IIUM.

By signing this form, I acknowledged that I have read and understand the IIUM Intellectual Property Right and Commercialization policy.

Affirmed by Hasan Khair bin Adzman

……..…….. 19 August 2025

Signature ……..
Date

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my supervisor, Dr. Raini Binti Hassan, for her consistent support, kindness, and timely guidance throughout the course of this research. Her constructive feedback, valuable suggestions, and insightful questions have played a crucial role in shaping this thesis. Dr. Raini's deep understanding of the research objectives and her ability to provide thoughtful input were particularly instrumental. Despite her numerous responsibilities, she always made herself available to provide assistance whenever needed, for which I am immensely grateful. I am also deeply appreciative of her moral support, which was a source of great encouragement during the drafting and finalization of this work.

I extend my sincere thanks to my co-supervisor, Ts. Dr. Dini Oktarina Dwi Handayani, for her steadfast support and collaboration, which were vital to the successful completion of this research.

I am deeply thankful to my parents and family for their unwavering prayers, patience, and understanding during this time.

Finally, I offer my utmost gratitude to Allah for His boundless mercy and blessings, which have enabled the completion of this thesis. Alhamdulillah.

This research was supported by the Fundamental Research Grant Scheme (FRGS) under Reference Code FRGS/1/2021/ICT01/UIAM/02/2, provided by the Ministry of Higher Education (MOHE) Malaysia.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GAT	Graph Attention Network
GCN	Graph Convolution Network
GNN	Graph Neural Networks
MCST	Master of Computer Science and Technology
ML	Machine Learning
OL	Online Learning
RL	Reinforcement Learning
XGBoost	Extreme Gradient Boosting

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

The Skyline query processing is widely used in multi-criteria decision-making applications such as route planning, product recommendation, and health diagnostics (Papadias et al., 2005). However, existing skyline methods face major challenges when applied to large and incomplete datasets, conditions that are increasingly common in real-world scenarios.

This work introduces a hybrid approach that combines Graph Convolutional Network (GCN) (Kipf and Welling, 2017) with the ISkyline dominance graph technique (Khalefa et al., 2008) to enhance skyline query performance. The proposed method is designed to handle missing data and scale efficiently, allowing for improved prediction of skyline tuples even in complex, incomplete environments. Experimental results on both synthetic and real-world datasets demonstrate that this method outperforms state-of-the-art techniques in accuracy and efficiency.

Skyline queries aim to retrieve data records that are not dominated by any others across multiple dimensions, often referred to as Pareto-optimal points. This concept has been extensively studied in database systems and multi-criteria optimization literature, where skyline operators are used to extract Pareto-optimal data points for decision support applications (Borzsony et al., 2001). While powerful, these queries are computationally expensive, especially when applied to massive graph-based data or datasets with incomplete attributes. Existing solutions like ISkyline and SIDS attempt to address scalability, but they still struggle with prediction under uncertainty or data loss.

Recent advances in deep learning, particularly Graph Convolutional Network (GCN), offer promising capabilities for learning from structured and incomplete data.

By integrating GCNs into the skyline processing workflow, the proposed method leverages graph-based feature learning to support more robust and intelligent skyline selection.

This research contributes to the development of more adaptive skyline frameworks and aligns with Sustainable Development Goal 3 (Good Health and Well-being) by enabling more informed decision-making from health-related graph data.

1.2 PROBLEM STATEMENT

Skyline queries are essential for identifying optimal data points from multi-dimensional datasets based on dominance relationships. However, traditional skyline query algorithms face significant limitations in processing large-scale and incomplete graph datasets. These methods often encounter challenges related to scalability, computational overhead, and inefficiencies in handling dynamic database environments (Wang et al., 2017; Khalefa et al., 2008). Furthermore, approaches like Bucket and ISkyline struggle with integrating missing data effectively, resulting in suboptimal accuracy and high processing costs (Bharuka and Kumar, 2013b). Current solutions lack a comprehensive framework that integrates cutting-edge advancements in machine learning, particularly Graph Convolutional Network (GCN), which offers the potential to address these challenges by improving scalability, accuracy, and adaptability (Afifi et al., 2024).

Despite the promising capabilities of machine learning, including its ability to model complex relationships in graph-structured data, its application in optimizing skyline queries remains underexplored (Mohamud et al., 2023). Existing research does not adequately leverage the dynamic adaptability and efficiency of machine learning techniques, leaving a critical gap in addressing the computational and data-handling shortcomings of traditional methods. This study aims to bridge these gaps by introducing a novel framework that combines Pareto optimality principles with advanced machine learning methods, offering robust solutions for scalable and efficient skyline computations in real-world scenarios.

1.3 PURPOSE OF THE STUDY

The aim of this research is to develop innovative and efficient methods for processing skyline queries by utilizing cutting-edge machine learning approaches, with a focus on Graph Convolutional Network (GCN). This research seeks to tackle fundamental challenges, including scalability, adaptability in dynamic environments, and handling incomplete data, which limit the performance of traditional algorithms. By creating a unified framework that combines Pareto optimality with machine learning, the study aims to improve the accuracy, efficiency, and real-world applicability of skyline queries, contributing to advancements in decision support systems, big data analytics, and dynamic database management.

1.4 RESEARCH OBJECTIVES

The study set out to accomplish the following objectives:

1. To develop a unified framework that integrates Pareto optimality principles with advanced machine learning techniques, particularly Graph Convolutional Network (GCNs), to improve skyline query processing over large-scale and attribute-incomplete graphs.
2. To evaluate the performance of the proposed framework across real-world and synthetic datasets using comprehensive metrics, including accuracy (target > 99%), F1-score (target > 99%), AUC-ROC (target > 99%), query response time, and memory usage, with a focus on ensuring scalability, efficiency, and adaptability in dynamic environments.
3. To compare the effectiveness of the proposed framework against traditional skyline algorithms and alternative machine learning models in order to identify the most suitable method for skyline query processing on incomplete graph-structured data.

1.5 RESEARCH QUESTIONS

1. How can Pareto optimality principles be effectively integrated with machine learning techniques to enhance skyline query performance?
2. How do the methods perform across a variety of real-world and synthetic datasets in terms of accuracy, F1-score, AUC-ROC, query response time, and memory usage?
3. Which method demonstrates the highest overall effectiveness for skyline query processing on incomplete graph-structured datasets, when evaluated using accuracy, F1-score, AUC-ROC, query response time, and memory usage?

1.6 RESEARCH HYPOTHESES

H1. The proposed GNN + ISkyline framework significantly improves skyline query classification accuracy compared to traditional skyline algorithms (e.g., ISkyline, SIDS, and OIS) when applied to incomplete graph-structured datasets.

H2. The integration of Graph Neural Networks (GNNs) with the ISkyline algorithm yields superior performance in skyline query classification on incomplete graph-structured datasets, as compared to other machine learning methods (e.g., GAT, XGBoost, Reinforcement Learning, and Online Learning), based on evaluation metrics such as accuracy, F1-score, and AUC-ROC.

1.7 SIGNIFICANCE OF STUDY

This study contributes to both academic research and practical applications by proposing a novel framework that combines Pareto optimality with Graph Convolutional Network (GCN) for efficient skyline query processing. Traditional algorithms often fall short when handling large, dynamic, and incomplete datasets. By

contrast, the proposed method leverages the learning capabilities of GCN to improve prediction accuracy and adaptability in complex graph environments.

Practically, this approach has broad applicability in domains such as e-commerce, finance, and smart infrastructure, where decisions must be made using incomplete and multi-dimensional data. The model also supports real-time systems by enabling scalable, responsive skyline computations. Furthermore, the research aligns with Sustainable Development Goal 9 by advancing intelligent data processing for innovation and infrastructure.

1.8 LIMITATIONS OF THE STUDY

Handling incomplete data poses a significant challenge in this research. While the study proposes leveraging machine learning techniques to manage datasets with missing values, accurately imputing or processing this incomplete information remains difficult, especially when the degree of incompleteness is high.

Scalability is another important concern. Although the research emphasizes scalable solutions, the application of machine learning models to very large datasets can still encounter bottlenecks. These issues often arise from high memory usage and extended processing times, particularly during the training phase of the models.

The central focus of the study is on optimizing skyline query processing over incomplete graph datasets. To achieve this, the research utilizes advanced machine learning techniques, with a particular emphasis on Graph Convolutional Network (GCN), which is well-suited for handling the complexities of graph-structured data.

1.9 SCOPE OF STUDY

This study focuses on developing and evaluating a hybrid framework that integrates Pareto optimality principles with Graph Convolutional Network (GCN) to improve skyline query processing over large-scale and attribute-incomplete graph-structured datasets. The scope includes both synthetic and real-world datasets, where data incompleteness is introduced at the attribute level while the graph structure remains

intact. The research compares the proposed GNN + ISkyline method against traditional skyline algorithms and machine learning models using evaluation metrics such as accuracy, F1-score, AUC-ROC, query response time, and memory usage. While the study emphasizes adaptability, efficiency, and robustness, it does not explore multi-hop graph reasoning, reinforcement learning, or structural graph incompleteness.

1.10 DEFINITIONS OF TERMS

Skyline Queries

An object is said to dominate another when it matches or exceeds the performance of the other across all dimensions and outperforms it in at least one. This principle is commonly used to determine the most optimal or preferred objects, known as skylines, within a dataset. These skylines are selected based on predefined user preferences, as specified in the context of skyline queries (Chomicki et al., 2004).

Incomplete Graphs

A database is classified as incomplete if it contains at least one data entry with missing values in one or more of its dimensions (Haas et al., 2002).

Machine Learning

Machine learning, a subset of artificial intelligence, involves designing algorithms and statistical models that enable computers to perform tasks without explicit programming (Mitchell, 1997). In essence, it utilizes statistical techniques to allow machines to learn from data and improve their performance over time.

Pareto Optimality

Pareto optimality, often referred to as Pareto efficiency, is an economic and decision theory principle where a state is achieved in which improving one individual's condition is impossible without disadvantaging at least one other individual (Mas-Colell et al., 1995).

1.11 CHAPTER SUMMARY

This study focuses on overcoming the limitations of traditional methods in processing skyline queries, which identify optimal data points for decision-making and analytics. Existing approaches struggle with scalability, dynamic adaptability, and handling incomplete data. This research proposes leveraging Graph Convolutional Network (GCN) and other machine learning techniques to create a unified framework that improves efficiency, scalability, and adaptability for skyline queries. Key goals include developing benchmarking performance across real-world and synthetic datasets. While promising significant advancements, the study acknowledges limitations such as challenges with highly incomplete data and scalability during model training.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

Skyline queries have become essential in multi-criteria decision-making and data filtering tasks due to their ability to identify non-dominated records based on user-defined preferences. However, the performance of traditional skyline algorithms degrades significantly when applied to incomplete or large-scale datasets. This chapter presents a detailed literature review that evaluates widely used skyline query methods, with particular emphasis on their strengths, limitations, and applicability to incomplete databases. A more thorough and focused analysis is provided to better define the research problem and highlight the need for a new integrated approach.

2.2 SKYLINE QUERIES

The skyline operator proposed by Borzsony et al. (2001), filters a dataset by selecting only those objects that are not dominated by any others. One object is said to dominate another if it is equal or better in all dimensions and strictly better in at least one. This method is widely used to extract the most preferred or optimal objects, referred to as skyline points, based on user-defined preferences, specified as a skyline query (Mohamud et al., 2023).

Skyline queries have become a significant focus in the database research community, particularly for applications involving multi-criteria decision-making. Since the introduction of the skyline operator by Borzsony et al. (2001), numerous algorithms have been developed to enhance skyline computation efficiency (Tan et al., 2001; Kossmann et al., 2002; Chomicki et al., 2004). These algorithms vary based on several factors, including: (i) the types of data they process, such as uncertain,

incomplete, encrypted, streaming, or big data; (ii) the computational platforms they utilize, such as distributed systems, cloud computing, or road networks; and (iii) the kinds of skyline queries they handle, including range skyline, spatial skyline, or reverse skyline queries (Mohamud et al., 2023).

The skyline operator functions by filtering objects in a dataset to select only those that are not dominated by others. In this context, an object is said to dominate another if it performs equally well or better across all dimensions and exceeds the other in at least one dimension. This mechanism helps identify the most preferred objects, referred to as skylines, which align with the user's specified preferences in a skyline query. Such an approach proves particularly useful for analyzing large-scale datasets, including those in graph databases (Mohamud et al., 2023).

A useful way to illustrate skyline queries is through a hotel database that includes details like room rates and proximity to the beach. Suppose a user wants to find hotels that are low-cost and located near the beach. A skyline query would retrieve only those hotels that are not outperformed by others in both price and distance, meaning no alternative offers a lower price and a closer location simultaneously (Saad et al., 2021).

The efficiency of skyline computation depends significantly on two factors: the number of dimensions and the size of the dataset. Additionally, the number of dimensions has a substantial impact on the query's search space. When the number of dimensions increases, the search space expands considerably, making it more challenging for objects to dominate one another (Yiu & Mamoulis, 2007).

The literature lacks comprehensive evaluation frameworks that benchmark the performance of skyline query algorithms across diverse datasets, particularly real-world and synthetic data. Metrics like query response time and memory usage are often overlooked or inconsistently reported.

2.3 SKYLINE QUERIES OVER LARGE SCALE GRAPHS

Wang et al. (2017) explored methods for processing skyline queries in large, incomplete databases and proposed an approach called Skyline Preference Query (SPQ). This method involves three main stages. First, the incomplete database is divided into two subsets based on the priority levels of attributes. The skylines of the first subset, referred

to as local skylines, are determined using the Skyline Incomplete Data Sets (SIDS) concept introduced by Bharuka and Kumar (2013a). Second, a bitmap representation technique is combined with the divide-and-conquer (DC) strategy from Borzsony et al. (2001) to identify the skylines of the second subset. Finally, the local skylines from both subsets are compared to produce the overall skyline of the database. However, SPQ requires generating multiple arrays and performing sequential processing, which involves exhaustive pairwise comparisons. This approach increases processing time because unnecessary comparisons are often conducted while identifying local skylines within each subset (Gulzar et al., 2021).

Processing skyline queries on massive datasets poses additional challenges due to the large number of candidates and the high computational cost of pairwise comparisons. Sorted-based algorithms address this by leveraging pre-sorted structures to select tuples with high dominance potential, thereby pruning non-skyline tuples. However, this method requires multiple passes over the dataset, leading to high input/output (I/O) costs, especially with large datasets. In the context of incomplete skyline computation, the criteria for dominance are broader than for complete datasets, resulting in a larger number of skyline candidates. Bucket-based algorithms, often used for this purpose, require significant resources due to the high number of buckets and local skyline results. These processes, particularly the computation and merging of local skyline results, incur substantial computational and I/O costs. As a result, existing algorithms struggle to efficiently handle incomplete skyline queries on massive datasets (He & Han, 2022).

The reviewed literature frequently emphasizes scalability as a challenge, particularly when processing skyline queries on large-scale datasets. However, existing algorithms either rely on exhaustive pairwise comparisons or lack mechanisms for efficient pruning, which restricts their applicability to massive datasets.

2.4 SKYLINE QUERIES OVER INCOMPLETE GRAPHS

Khalefa et al. (2008) introduced two algorithms, Bucket and ISkyline, for addressing the issue of skyline queries on incomplete data. The Bucket algorithm uses a bitmap representation to divide database tuples into distinct buckets, where each bucket

contains tuples with similar missing attributes. A conventional skyline algorithm is then applied to each bucket to identify local skylines. These local skylines are compared across buckets to determine the global skylines of the entire database. The ISkyline algorithm improves upon this by introducing optimization techniques such as virtual points and shadow skylines, which reduce the number of local skylines in each bucket. This reduction minimizes pairwise comparisons, though the use of virtual points increases computational overhead due to additional comparisons (Gulzar et al., 2021).

Arefin and Morimoto (2012) proposed the Replacement-Based Sets Skyline Queries (RBSSQ) method to handle skyline queries in datasets with missing values. This approach builds on the Bucket algorithm by replacing missing values with numbers larger than the attribute domain, preserving the transitivity property of skyline queries and avoiding cyclic dominance. RBSSQ operates in two phases: data preprocessing, where missing values are replaced, and skyline sets computation, where the processed data is used to compute the skylines (Gulzar et al., 2021).

Bharuka and Kumar (2013a) proposed the Sort-based Incomplete Data Skyline (SIDS) algorithm, which adapts a sorting mechanism for skyline computation as outlined by Balke et al. (2004). SIDS processes tuples round-robin by attributes, pruning dominated tuples early to minimize pairwise comparisons. If a tuple remains unpruned after k iterations, where k is the count of its complete attributes, it is deemed a skyline. While efficient in sequential access, SIDS faces performance challenges with increasing attribute lists and lacks optimization for databases with dynamic content. This limitation makes it less suitable for systems requiring real-time updates (Gulzar et al., 2021).

Bharuka and Kumar (2013b) later introduced the Incomplete Data Frequent Skyline (IDFS) algorithm, leveraging the top-k frequent skyline technique proposed by Chan et al. (2006). IDFS identifies superior skylines based on their fractional frequency in the database. While effective in controlling skyline size, IDFS must be reapplied when the database content changes, limiting its efficiency for dynamic databases (Gulzar et al., 2021).

Miao et al. (2013) proposed two algorithms, k-iSkyband (kISB) and Virtual Point-based (VP), to optimize skyline queries on incomplete data. kISB employs concepts such as thickness warehouses and expired skylines to enhance query performance for k-Skyband queries. The VP algorithm uses virtual points, expired skylines, and shadow skylines to reduce the size of candidate sets and improve query

efficiency. These methods are inspired by the earlier work of Khalefa et al. (2008) (Gulzar et al., 2021).

Alwan et al. (2016) developed the Incoskyline algorithm to address cyclic dominance and the loss of transitivity in skyline computations for incomplete data. The method consists of four phases: clustering, local skyline identification, k-dom skyline generation, and final skyline computation. Virtual tuples, or k-dom skylines, are used to prune dominated tuples, and candidate skylines are compared to produce the final results. However, Incoskyline is not well-suited for dynamic databases, as it requires a complete reapplication for updates, involving rescanning and exhaustive pairwise comparisons (Gulzar et al., 2021).

The Optimized Incomplete Skyline (OIS) framework comprises four main components: i) clustering, ii) sorting and filtering, iii) local skyline identification, and iv) final skyline identification. Its primary goal is to streamline the skyline computation process in databases containing incomplete data. This simplification focuses on minimizing the number of dominance comparisons required to identify skyline points. However, OIS lacks support for missing data and real-time adaptability (Gulzar et al., 2019).

Current methodologies for handling incomplete data, such as Bucket and Iskyline, address the issue of missing values but often result in high computational overhead. Few approaches consider leveraging machine learning to enhance the efficiency and accuracy of skyline computations in this context.

Many existing algorithms, such as Incoskyline and SIDS, are designed for static databases. These approaches are inadequate for dynamic environments where database contents frequently change due to updates, insertions, or deletions.

2.5 LINK BETWEEN MACHINE LEARNING AND SKYLINE QUERIES

Artificial Intelligence (AI) encompasses systems or machines capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, language understanding, and decision-making. Machine Learning (ML), a subset of AI, focuses on developing algorithms and statistical models that enable computers to learn from data and improve their performance over time

without being explicitly programmed. By employing statistical methods, ML systems excel in generating high-quality outputs, such as classifications, regression values, and generated artifacts, even in complex environments with intricate decision boundaries. For these challenging scenarios, ML models can significantly reduce computational resources needed for adequate responses and often deliver superior outcomes compared to earlier models. However, more complex problems demand substantial computing power and large amounts of training data. Since ML models rely heavily on generalizing from data records, the quality of these data samples is crucial for overall model effectiveness (Afifi et al., 2024).

AI involves machines solving problems by perceiving their environment and leveraging a knowledge model to derive solutions and make decisions. ML is a fundamental component of AI and is considered one of its primary subfields (Russell, 2010). Unlike explicitly programmed systems, ML models use historical data or experience as evidence to build statistical and computational frameworks for performing tasks. ML methods are generally categorized into three paradigms: supervised learning, unsupervised learning, and reinforcement learning (RL). These paradigms differ based on the type of feedback provided to the learning system. Supervised learning uses labelled data for precise feedback, whereas unsupervised learning involves partially labelled or entirely unlabelled data. In RL, the system receives implicit feedback through a reward function, assigning numerical values to observed data based on performance (Afifi et al., 2024). Machine learning has a broad range of applications, including cybersecurity (Halbouni et al., 2022), fraud detection (Wang et al., 2022; Alarfaj et al., 2022), and stroke identification (Saleem et al., 2024).

Although some studies explore the use of machine learning in skyline queries, there is a notable scarcity of research applying graph convolutional network (GCN) to optimize skyline queries over large-scale and incomplete graphs. Existing works predominantly. rely on conventional algorithms, which may lack scalability and adaptability to dynamic datasets.

2.6 GRAPH CONVOLUTION NETWORK (GCN)

Graph Convolutional Networks (GCNs) have become a foundational model for learning over graph-structured data, particularly in tasks such as node classification, link prediction, and graph-level classification. Introduced by Kipf and Welling (2017), GCNs efficiently generalize the convolution operation to non-Euclidean domains by leveraging spectral graph theory and neural network techniques. Their approach formulates convolution as a layer-wise propagation rule that combines node features with their local graph structure, enabling the model to capture both attribute and topological information. This innovation has since sparked extensive research and development in graph convolutional network across various domains, including social networks, biology, recommendation systems, and knowledge graphs.

GCN architecture consists of three graph convolutional layers, each responsible for aggregating and transforming node features based on their local graph neighborhoods. The input to the network is a node feature matrix $X \in \mathbb{R}^{N \times F}$, where N is the number of nodes and F is the number of input features per node. Each graph convolutional layer performs a neighborhood aggregation as described by Kipf and Welling (2017):

$$H^{l+1} = \sigma\left(\widehat{D}^{-1/2}\hat{A}\widehat{D}^{-1/2}H^{(l)}W^{(l)}\right) \quad (1)$$

where $H^{(l)}$ is the node representation at layer l, $\hat{A} = A + I$ is the adjacency matrix with added self-loops, $\widehat{D}$ is the degree matrix, $W^{(l)}$ is the learnable weight matrix, and σ is an activation function (commonly ReLU). This operation ensures that each node's updated representation reflects both its own features and those of its neighbors, enabling the model to learn relational patterns in graph-structured data efficiently. GCNs are particularly effective for node classification tasks and have been successfully applied to problems involving sparse, incomplete, or partially labelled data.

2.7 GRAPH NEURAL NETWORKS (GNNS)

According to Afifi et al. (2024), Graph Neural Networks (GNNs) have recently become highly effective for managing graph-structured data. These networks utilize permutation-invariant aggregation or pooling methods along with permutation-

equivariant message-passing techniques to identify patterns in the data, ensuring the graph's topology is maintained without requiring a specific arrangement of its nodes and edges (Veličković, 2023).

GNNs are highlighted as powerful tools for handling graph-structured data due to their ability to preserve the graph's topology while identifying patterns through permutation-invariant aggregation and message-passing techniques. This capability ensures that the structure of the graph is maintained regardless of node or edge arrangement, making GNNs particularly effective for tasks requiring an understanding of complex relationships within graph data.

2.8 REINFORCEMENT LEARNING

Reinforcement learning involves determining the best course of action by associating situations with actions to maximize a numerical reward. Rather than being explicitly instructed on which actions to take, the learner must explore and identify the actions that yield the highest rewards through experimentation. In complex scenarios, actions can influence not only the immediate reward but also the subsequent state and, consequently, future rewards. The two defining aspects of reinforcement learning are its reliance on trial-and-error exploration and the concept of delayed rewards (Sutton & Barto, 2018).

2.9 ONLINE LEARNING

The traditional machine learning approach typically operates in a batch learning mode, such as in supervised learning tasks, where a model is trained using a predefined dataset with a specific learning algorithm. This approach requires the complete training dataset to be available beforehand, and the training process is generally performed offline due to its high computational cost. However, batch learning methods face significant limitations, including (i) inefficiency in terms of time and space usage and (ii) limited scalability for large-scale applications, as the model must often be retrained from the beginning when new data becomes available. In contrast, online learning processes data

sequentially, allowing the model to learn and adjust predictions continuously with each new data instance. This method addresses the drawbacks of batch learning by enabling instant updates to the predictive model, making it much more efficient and scalable for real-world scenarios where data is both large and generated at a rapid pace (Hoi et al., 2021).

2.10 COMPARISON OF MACHINE LEARNING TECHNIQUES

Table 2.1 Comparison of Machine Learning Techniques

Technique	Strengths	Limitations	Best Use Cases
Graph Neural Networks (GNNs) (Scarselli et al., 2009)	- Captures relationships in graph-structured data. - Handles both local and global graph features effectively. - Scales well with parallelizable message-passing techniques.	- High computational cost for very large graphs. - Requires expertise in graph theory and GNN design.	- Social network analysis, recommendation systems, and multi-dimensional data with relational contexts.
Graph Attention Networks (GATs) (Veličković et al., 2018)	- Uses attention mechanisms to prioritize important graph relationships. - Enhances interpretability by focusing on	- Computationally expensive compared to other GNN variants. - Struggles with scalability for extremely large graphs.	- Scenarios requiring weighted importance of graph nodes or edges, like traffic prediction or fraud detection.

	influential graph components.		
XGBoost (Chen & Guestrin, 2016)	- High-speed, scalable, and accurate gradient boosting. - Robust against overfitting with regularization techniques.	- Limited performance for graph-structured or highly relational datasets. - Requires significant feature engineering for high-dimensional data.	- Tabular data, ranking problems, and structured datasets with clear feature engineering.
Reinforcement Learning (RL) (Sutton & Barto, 2018)	- Learns adaptive policies through trial and error. - Effective for problems without explicit supervision. - Excellent for real-time decision-making tasks.	- High computational cost and slower convergence in complex environments. - Requires well-defined reward functions, which can be challenging to design.	- Dynamic environments like robotics, game AI, and dynamic skyline query systems.
Online Learning (OL) (Crammer et al., 2006)	- Adapts to new data in real-time without retraining. - Efficient for dynamic and evolving datasets.	- Limited to simple algorithms or requires incremental versions of complex models. - May struggle with catastrophic forgetting in non-	- High-velocity data streams like IoT systems, financial trading, and recommendation engines.

		stationary environments.	
GraphSAGE (Hamilton et al., 2017)	- Generalize to unseen nodes or graphs. - Compatible with mini-batch stochastic gradient descent.	- Neighbor sampling can miss important structural or semantic information. - Aggregation is based only on local neighborhoods.	- Dynamic or evolving networks where new nodes and edges are continuously added.

2.11 PARETO OPTIMALITY

Skyline queries, grounded in the concept of Pareto dominance, are designed to filter objects from a potentially large multi-dimensional dataset by selecting those that best align with user preferences. These queries retain the most favorable objects while disregarding others that do not meet the criteria (Dehaki, Ibrahim, Alwan, et al., 2021).

The concept of Pareto optimality originated from economic equilibrium and welfare theories in the early 20th century. This principle states that a system achieves optimal efficiency when it is impossible to improve one individual's condition without negatively impacting another's. Named after Italian economist Vilfredo Pareto (1848–1923), this concept has since become a cornerstone in economics and has found applications across disciplines, including social sciences, engineering, management, and information systems (Luc, 2008).

A formal mathematical framework for Pareto optimality was introduced by Kuhn and Tucker (1951), who developed necessary and sufficient conditions for efficiency. Around the same time, Koopmans (1951) applied the concept to operations research. Subsequent advancements were made by researchers such as Zadeh (1963), Klinger (1964), Da Cunha and Polak (1967), and Geoffrion (1968). Despite these early contributions, significant developments in Pareto optimality theory and its applications began in earnest during the 1970s and 1980s. Today, the concept has inspired extensive

research, with thousands of papers and books dedicated to advancing both theoretical insights and practical applications (Luc, 2008).

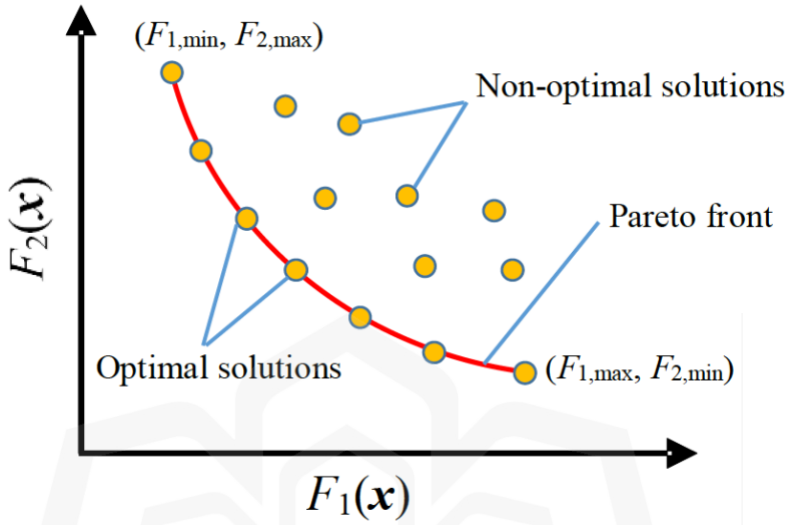


Figure 2.1 Pareto-Optimal Solutions

Source: (Mergos and Sextos, 2018)

The image illustrates the concept of the Pareto front in multi-objective optimization, where optimal trade-offs between two competing objectives $F_1(x)$ and $F_2(x)$ are visualized. Solutions along the red curve represent non-dominated, optimal solutions forming the Pareto front, while the scattered yellow points above it indicate non-optimal, dominated solutions. This visualization is adapted from the work of Mergos and Sextos (2018), where they employed genetic algorithms to achieve balanced seismic performance objectives by exploring such Pareto-optimal solutions.

The reviewed studies lack a unified framework that combines the strengths of Pareto optimality with advanced machine learning techniques. A cohesive framework addressing both scalability and incompleteness in datasets is missing from current research.

2.12 SUMMARY OF COMMONLY USED METHODS

The most commonly used methods in skyline query processing include ISkyline, Bucket, RBSSQ, Incoskyline, SIDS, IDFS, k-iSkyband/VP, and OIS. ISkyline uses bitmap representations and virtual points to reduce pairwise comparisons in datasets with missing values. It performs well under moderate incompleteness but incurs high memory overhead, particularly when the data has numerous virtual points. The Bucket algorithm partitions data with similar missing patterns into manageable groups, allowing local skyline computations within each bucket. Although it improves performance over brute-force methods, its computational cost rises significantly with an increase in the number of buckets or frequent updates.

RBSSQ replaces missing values with values beyond the known domain to preserve transitivity and dominance rules. While it maintains mathematical consistency, the artificial values may distort skyline results and impact query reliability. Incoskyline applies clustering and k-dom skyline identification, offering a more intuitive structure for handling incomplete data. However, it requires complete reprocessing after updates, limiting its usefulness in dynamic environments (Arefin and Morimoto, 2012). SIDS leverages sorting and sequential pruning of tuples, offering faster skyline extraction in static datasets, though it struggles with high dimensionality and incomplete records (Bharuka and Kumar, 2013a).

IDFS identifies skylines based on their frequency of dominance and has proven useful in limiting skyline size. Still, it lacks real-time adaptability as it must be fully re-executed after each data change (Bharuka and Kumar, 2013b). The k-iSkyband and Virtual Point (VP) methods reduce candidate sets by using shadow skylines and expiration rules. These techniques are particularly helpful in limiting the comparison overhead in sparse datasets but remain complex to implement at scale (Miao et al., 2013).

A notable method, OIS (Optimized Input/Output Skyline), has also been introduced to minimize I/O overhead in skyline processing. OIS is designed for systems that handle large-scale datasets and are constrained by memory or disk performance. It uses optimized sorting and data access techniques to reduce I/O bottlenecks, making it suitable for resource-constrained environments. Its primary strength lies in reducing query time through efficient disk access strategies. However, OIS is limited in handling

missing values and does not inherently support dynamic updates or learning-based improvements, making it less adaptable for evolving data environments (Gulzar et al., 2019).

To provide a structured overview, Table 2.1 presents a summary of the most widely used methods in skyline query research, highlighting their key operational features, strengths, and limitations.

Table 2.2 Summary of Common Skyline Query Methods and Their Characteristics

Method	**Description**	**Strengths**	**Limitations**
ISkyline (Khalefa et al., 2008)	Uses buckets and virtual points to reduce comparisons.	Efficient for moderate incompleteness; reduces pairwise comparisons.	High memory overhead; not ideal for dynamic or large datasets.
Bucket Algorithm (Khalefa et al., 2008)	Organizes tuples with similar missing values into buckets.	Simplifies local skyline computation.	High computational cost when bucket count is large; lacks adaptability.
RBSSQ (Arefin and Morimoto, 2012)	Replaces missing values with values outside domain.	Preserves transitivity and avoids cyclic dominance.	Risk of distortion; does not scale well.
Incoskyline (Alwan et al., 2016)	Uses clustering and k-dom skylines for incomplete data.	Prunes dominated tuples; intuitive clustering.	Full recomputation required after updates; limited dynamic support.

SIDS (Bharuka and Kumar, 2013a)	Sorts and prunes tuples using round-robin strategy.	Fast for static and moderately incomplete data.	Weak performance in highly incomplete or dynamic datasets.
IDFS (Bharuka and Kumar, 2013b)	Selects frequent skylines based on occurrence count.	Controls skyline size; effective for top-k results.	Inefficient in dynamic datasets; requires reprocessing.
k-iSkyband / VP (Miao et al., 2013)	Uses shadow skylines and expiration rules.	Reduces candidate set size effectively.	Complex structure; struggles with high incompleteness.
OIS (Gulzar et al., 2019)	Optimizes skyline I/O performance with efficient data access.	Reduces I/O cost in large-scale environments; suitable for disk-bound systems.	Lacks support for missing data and real-time adaptability.

2.13 COMPARISON OF MISSING DATA HANDLING APPROACHES

2.13.1 Importance of Handling Missing Data in Skyline Queries

Preserving accuracy in decision-making is a key concern when dealing with missing data in skyline queries. These queries are designed to identify optimal data points by comparing objects across multiple dimensions based on user preferences. However, when data is missing, it can disrupt the accuracy of dominance relationships, leading to incomplete or incorrect results. For instance, in a dataset evaluating hotels based on price and distance from the beach, missing price values could result in an inferior hotel being incorrectly classified as part of the skyline, thereby skewing the outcome. This challenge has been widely discussed in the literature, particularly in contexts where decision-making accuracy is critical despite incomplete data (Gulzar et al., 2021).

The presence of missing data also impacts the principle of Pareto optimality, which underpins skyline queries. According to this principle, an object is considered superior if it performs better in at least one dimension without being worse in others. Missing values complicate the assessment of dominance, making it difficult to determine whether an object truly belongs to the skyline.

Additionally, managing missing data introduces increased computational overhead (Khalefa et al., 2008). Techniques such as imputation, filtering, or algorithm redesign are often required, adding complexity and significantly raising the computational cost, particularly in large datasets. This overhead becomes more pronounced in dynamic and large-scale environments, where missing data is more common due to factors like real-time updates, data integration from diverse sources, or sensor failures. Efficient handling in such scenarios is crucial to maintain scalability and adaptability in skyline query processing.

Lastly, the challenge of missing data is especially critical in application domains such as decision support systems, big data analytics, and recommendation engines. In these areas, ensuring accurate and trustworthy results is vital, as decisions based on incomplete or inaccurate data can lead to negative outcomes. Therefore, robust strategies for handling missing data are essential to uphold the reliability of skyline queries in these critical contexts.

2.13.2 Why a Comparative Study is Needed

There is a notable diversity of existing approaches for handling missing data in skyline queries, ranging from traditional statistical methods like mean imputation to more sophisticated algorithms such as Iskyline and Incoskyline. Each of these techniques comes with its own set of strengths and limitations, making a comprehensive evaluation of their performance essential. However, a major gap in the current literature is the lack of unified frameworks that holistically compare these methods. Most studies focus on specific aspects such as efficiency or accuracy, but do not provide an all-encompassing analysis across key metrics like accuracy, scalability, and computational overhead (Mohamud et al., 2023).

In addition, the rapid advancement of machine learning, particularly with techniques like Graph Convolutional Network (GCN), opens up new possibilities for handling missing data more effectively (Kipf & Welling, 2017). A comparative study is necessary to benchmark these modern approaches against traditional methods and highlight their potential advantages. Such a study would also serve as a practical guide for real-world applications, offering clear recommendations on the most suitable approaches for various scenarios, such as static versus dynamic datasets or datasets with low versus high levels of incompleteness. Such benchmarking is necessary to reveal trade-offs and guide the design of more robust skyline systems (Mohamud et al., 2023).

Moreover, a thorough comparison of existing methods helps identify research gaps, revealing where current techniques may fall short, such as in adapting to dynamic environments or handling severe data incompleteness. These insights further justify the development of innovative solutions like the proposed GCN-based framework, which aims to address these limitations and enhance the robustness and scalability of skyline query processing.

2.13.3 Comparison of Approaches

Table 2.3 Comparison of Approaches

Method	**Accuracy**	**Scalability**	**Computational Overhead**	**Adaptability**	**Robustness to Incompleteness**
ISkyline (Khalefa et al., 2008)	High	Medium	High	Low	Medium
Bucket Algorithm (Khalefa et al., 2008)	Medium	Medium	Medium	Low	Medium
RBSSQ (Arefin and	Medium	Low	High	Low	High

Morimoto, 2012)					
Incoskyline (Alwan et al., 2016)	High	Low	High	Low	High
SIDS (Bharuka and Kumar, 2013a)	Medium	Medium	Medium	Low	Medium
IDFS (Bharuka and Kumar, 2013b)	Medium	Low	Medium	Low	Medium
k-iSkyband (Miao et al., 2013)	Medium	Medium	Medium	Low	Medium
OIS (Gulzar et al., 2019)	Medium	High	Low	Low	Low

In this table, accuracy reflects the method's ability to consistently return correct skyline results (higher than 90% for high, 70–90% for medium, lower than 70% for low), scalability assesses performance trends as data size increases; high scalability implies linear or near-linear performance trends, computational overhead measures the memory and processing cost relative to dataset size, adaptability indicates how well the method accommodates updates, insertions, or deletions in real-time, robustness to incompleteness refers to the method's performance when more than 20% of the dataset contains missing values. These characteristics are drawn from existing evaluations of traditional skyline algorithms under varying conditions (Bharuka and Kumar, 2013b).

These comparisons highlight a critical insight: no existing method effectively satisfies all the requirements for skyline query processing in large, incomplete, and dynamic datasets. ISkyline and Incoskyline offer strong accuracy but lack real-time

adaptability. OIS achieves low computational overhead and excellent scalability but fails to support incomplete or evolving datasets. Other methods are tailored for specific problem aspects but cannot generalize across varying data types or conditions.

This reinforces the need for an integrated approach that leverages the strengths of multiple techniques to overcome the limitations identified. The literature makes it evident that there is no one-size-fits-all method; hence, combining Graph Convolutional Network (for structure and imputation) and Pareto optimality (for accurate skyline logic), presents a promising and novel solution. The next chapter introduces this hybrid framework and justifies each component's inclusion based on the findings summarized here.

2.14 CHAPTER SUMMARY

The literature reveals a broad spectrum of skyline query algorithms developed to tackle the challenges posed by high-dimensional and large-scale datasets, especially when faced with missing or incomplete values. While traditional methods such as ISkyline, Bucket, and Incoskyline introduce optimizations through partitioning and pruning strategies, they often incur high computational costs and lack adaptability in dynamic environments. Although recent work acknowledges the promise of machine learning, particularly in handling complexity and data incompleteness, few approaches leverage graph-based learning models that can directly exploit structural relationships within data. This persistent gap motivates the exploration of hybrid frameworks that blend conventional dominance logic with advanced learning techniques.

In conclusion, based on this literature review, ISkyline emerges as the most balanced traditional method in terms of robustness and accuracy, but its limited adaptability and scalability underline the need for innovation. The limitations of all reviewed approaches justify the development of a unified, learning-driven framework for efficient skyline query processing in complex, real-world environments.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 INTRODUCTION

This chapter aims to detail the methodologies employed in this study, drawing from the design science research cycle as outlined by (Hevner, 2004). The approach integrates three cycles to enhance the identification and comprehension of design science research initiatives. Figure 3.1 illustrates the adapted design science research framework based on the work of (vom Brocke et al., 2020).

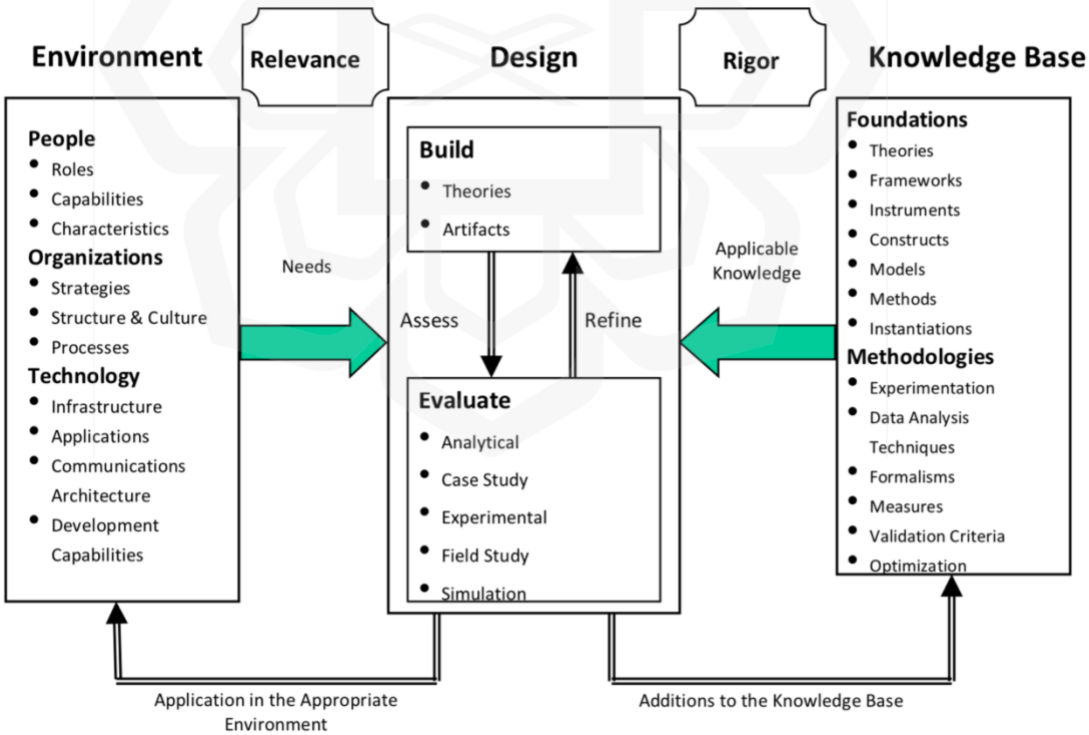


Figure 3.1 Design Science Research Framework

Source: (vom Brocke et al., 2020)

The Relevance Cycle connects the research project's contextual environment with design science activities, ensuring that the needs and requirements for achieving the research objectives are properly identified. The Rigor Cycle ties design science activities to a knowledge base of scientific principles, expertise, and prior experiences that inform and influence the research process (Hevner & Chatterjee, 2010). At the core is the Design Cycle, which focuses on developing and evaluating design artifacts and research processes. This cycle plays a pivotal role in describing the research activities. Figure 3.2 illustrates the sequential flow of the research process within this framework.

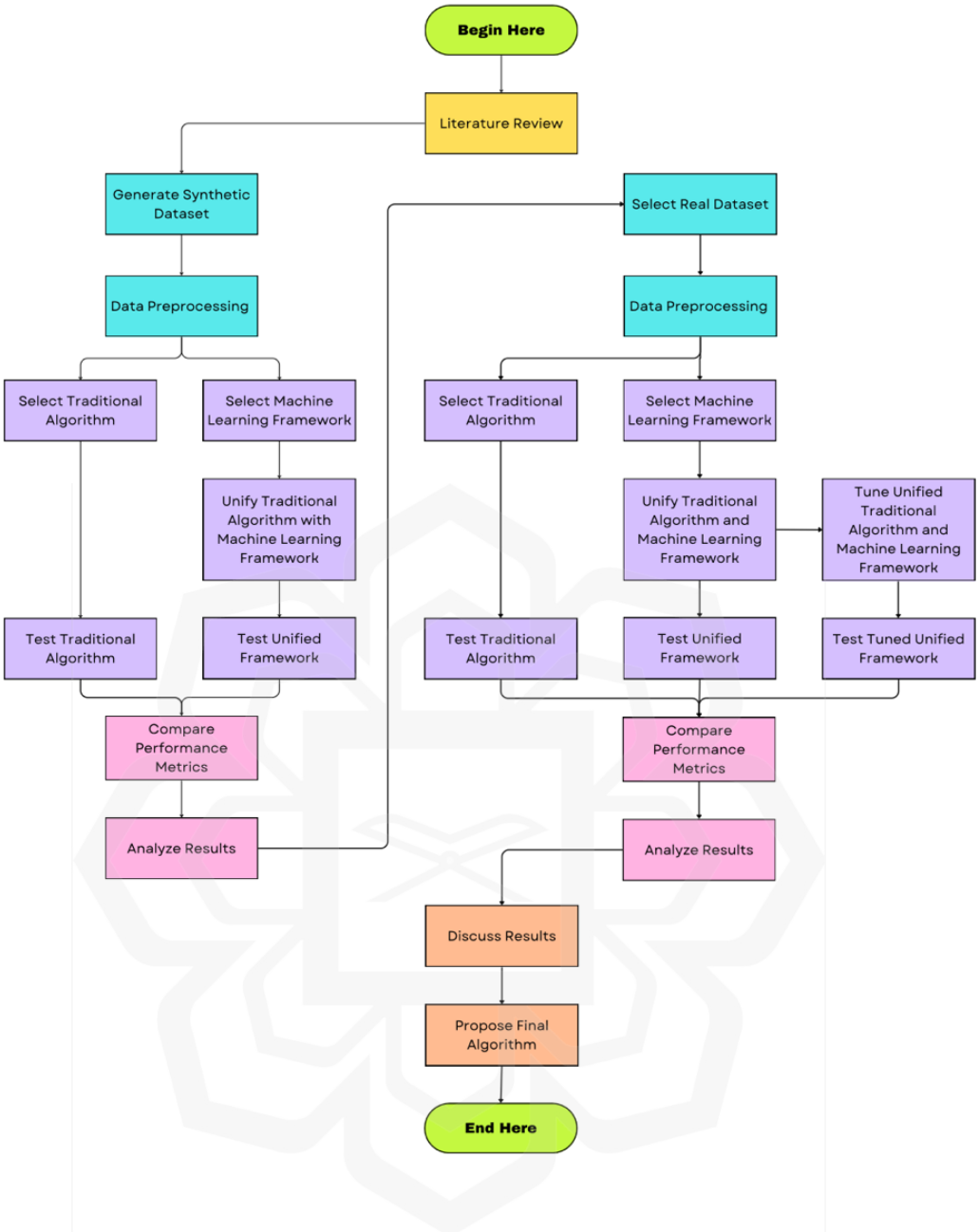


Figure 3.2 Design Cycle

The workflow begins with a literature review, providing a foundational understanding of traditional skyline algorithms, machine learning frameworks, and graph-based methodologies. The process then advances to testing synthetic datasets as the initial step, followed by real datasets. Both workflows incorporate data preprocessing, which includes tasks such as normalizing data, handling missing values,

and splitting datasets into training, validation, and test sets. For synthetic datasets, the research involves the selection of traditional algorithms (e.g., ISkyline) and machine learning frameworks (e.g., Graph Neural Networks), which are then unified to leverage the strengths of both approaches. The unified framework, along with traditional algorithms, is tested extensively, followed by a comparison of performance metrics such as processing time, memory usage, and accuracy. Results are analyzed to evaluate the effectiveness of the unified approach. Similarly, the real datasets testing follows the same workflow but includes an additional step of tuning the unified machine learning framework to optimize its performance for real-world applications. This tuning involves hyperparameter adjustments and testing alternative architectures (e.g., GraphSAGE). Finally, the results from both workflows are consolidated, discussed comprehensively, and used to propose a final algorithm that demonstrates superior performance in terms of scalability, robustness, and adaptability. This expanded explanation ensures a clear understanding of how each step in the methodology contributes to achieving the research objectives.

3.2 PROPOSED METHOD: GNN + ISKYLINE

This study proposes a hybrid method that integrates a traditional skyline query algorithm, ISkyline, with a Graph Neural Network (GNN), instantiated as a Graph Convolutional Network (GCN) (Kipf & Welling, 2017) to handle attribute-level incompleteness in graph-structured data. GCN was selected as the specific GNN model due to its efficiency in semi-supervised learning and ability to aggregate information from neighboring nodes in incomplete graphs. The goal is to learn dominance relationships between data points and improve skyline prediction in incomplete datasets through graph-based deep learning.

The methodology begins by collecting data. The data collection stage involves identifying, selecting, and gathering datasets relevant to the skyline query processing task. This includes both real-world datasets, such as CoIL 2000, NBA Stats, and MovieLens, and synthetic datasets designed to test the framework under controlled conditions.

Once the datasets are collected, the preprocessing phase ensures they are clean and ready for analysis. Numerical attributes such as price, rating, and availability are normalized to a common scale to prevent any single feature from dominating the learning process, while categorical variables are encoded into machine-readable formats. The processed datasets are then loaded into the analytical environment, where they are transformed into node–feature matrices and associated metadata structures suitable for graph-based modeling. Using the ISkyline algorithm, the ground truth skyline is computed by systematically comparing each data point against others to determine dominance relationships based on Pareto optimality.

A modified version of the ISkyline algorithm is implemented, which compares pairs of data points to determine dominance, taking into account missing values by skipping None comparisons. The dominates() function checks if one point dominates another by verifying that all comparable attributes are greater than or equal to and at least one attribute is strictly greater. This implementation follows the skyline dominance logic introduced in early skyline algorithms (Borzsony et al., 2001). The output of this process produces a binary label for each data point indicating whether it belongs to the skyline or not.

Following the skyline computation using the ISkyline algorithm, the Pareto Optimality Principle is applied to formalize the dominance relationships among tuples. This principle ensures that only data points that are not dominated by any other across all relevant dimensions are retained. A point is Pareto-optimal if no other point performs equally well or better in all dimensions and strictly better in at least one (Luc, 2008). Formally, for any two tuples p and q, p dominates q (denoted as $p \prec q$) if $p_i \leq q_i$ for all $i \in D$ and $p_j < q_j$ for at least one $j \in D$, where D represents the set of dimensions with non-missing values. This can be mathematically expressed as:

$$\mathrm{x} * \in S_p = \{\mathrm{x} \in \mathbb{R}^2 | \, 0 \leq x_1 \leq 2, x_2 = 0\} \quad (2)$$

This equation defines a subset of optimal solutions in $\mathbb{R}^2$ bounded by specific constraints, illustrating the condition where x_2 remains constant and x_1 lies within a defined range. Embedding such a formulation into the framework guarantees that the subsequent dominance graph construction reflects true multi-criteria optimality as defined by Pareto efficiency.

Once the skyline points are identified, the dataset is balanced by oversampling the skyline class to match the number of non-skyline points. These balanced samples

are used to construct a graph where nodes represent products for first synthetic dataset in subchapter 3.3.4 and edges represent dominance relationships based on the ISkyline method. Specifically, an edge is created from node *i* to node *j* if *i* dominates *j*. This yields a directed dominance graph from which a GNN can learn.

The GNN model architecture is a 3-layer Graph Convolutional Network (GCN) with dropout layers to mitigate overfitting. It takes the dominance graph as input, where node features are the normalized attribute values (e.g., Price, Rating, Availability, Shipping Time, and Category), and the labels are the binary skyline indicators. The model is trained using binary cross-entropy loss with logits and class imbalance adjustment via positive weight. The dataset is split into training (80%) and testing (20%) subsets using random masking.

Training is conducted for 200 epochs, and the model's performance is evaluated on multiple metrics, including accuracy, precision, recall, F1-score, and AUC-ROC. Additionally, two computational efficiency metrics, which are query response time and peak memory usage, were tracked using Python's performance monitoring tools.

Unlike traditional imputation methods that attempt to fill in missing values directly, the GCN in this framework does not impute missing data in a conventional sense. Instead, it learns to make accurate predictions by aggregating feature information from neighboring nodes through the graph structure (Kipf & Welling, 2017). This neighborhood-based propagation allows the model to compensate for missing attributes by leveraging correlated information from surrounding nodes that have complete or more reliable data. As such, the GCN implicitly addresses missingness during training, without modifying the original data values. This makes the framework particularly robust in scenarios with high levels of incompleteness, where direct imputation may introduce bias or noise.

This hybrid method is especially suited for datasets with attribute-level incompleteness, as it combines the decision logic of skyline dominance with the pattern-learning capabilities of GCN. Unlike traditional skyline methods that falter under incompleteness, the GCN learns to infer dominance patterns even when certain attribute values are missing, leveraging the structure of the dominance graph.

By integrating ISkyline and GNN in a single pipeline, this method addresses the limitations of purely algorithmic or purely learned approaches. It also enables

evaluation under synthetic and real-world datasets. Figure 3.3 below encapsulates the above steps and serves as a high-level overview of the proposed hybrid methodology.

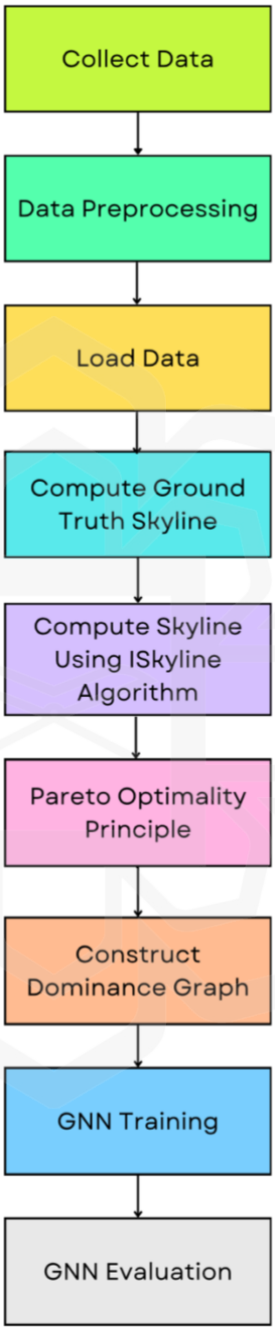


Figure 3.3 Proposed GNN + ISkyline Framework

By structuring the methodology in this modular fashion, the proposed approach offers transparency in its design while allowing each step to be independently assessed and improved. The integration of dominance relations with graph learning bridges classical skyline computation and modern neural representation learning, creating a more scalable and accurate solution for handling incomplete multi-criteria datasets.

The Graph Convolutional Network (GCN) was selected as the base model due to its proven effectiveness in learning from node features and local graph structure while maintaining computational efficiency. GCN performs well in scenarios with moderate graph sizes and achieves robust feature propagation through message-passing mechanisms, making it suitable for skyline dominance tasks (Kipf & Welling, 2017). Unlike attention-based models like GAT, which require higher computational overhead, GCN offers a good balance between performance and efficiency, especially critical when working with large and partially incomplete datasets (Veličković, 2023). Additionally, its simplicity enables easier integration and tuning within the proposed ISkyline hybrid framework.

3.3 DATASET PREPARATION

3.3.1 Real-World Datasets

To evaluate the proposed methods on diverse, large-scale, and incomplete graph datasets, this study utilizes real-world datasets such as the CoIL 2000 insurance dataset (Putten & Someren, 2000), NBA player statistics from Basketball Reference (Basketball Reference, n.d.), and the MovieLens rating dataset (Harper & Konstan, 2015).

The selection of these datasets is a critical step in ensuring that the proposed methods are tested under realistic and practical conditions. Each dataset aligns with the research objectives and offers specific advantages for testing various facets of the proposed approach. For example, CoIL 2000 provides customer information across multiple dimensions, making it suitable for evaluating multi-criteria decision-making and handling missing attributes in dynamic settings. NBA Stats contains real-world player performance metrics, which are inherently multi-dimensional and often incomplete, ideal for testing scalability and accuracy in skyline queries. MovieLens,

widely used in recommendation system research, includes user preferences and ratings, providing a relevant scenario for assessing the method's performance on large-scale, partially incomplete datasets.

The diversity of data characteristics across these datasets further enhances the evaluation. CoIL 2000 includes both categorical and numerical data, allowing the algorithm to be tested for its ability to handle mixed data types. NBA Stats, with dense and numerical attributes, challenges the method's scalability and efficiency in processing dominance relationships. In contrast, MovieLens contains sparse data that emphasizes the need for effective handling of incomplete entries and preference-based evaluations.

Moreover, these datasets vary in volume and complexity, which helps assess the robustness and scalability of the proposed method. CoIL 2000 is a small to medium-sized dataset with 5,822 nodes, while NBA Stats represents a medium-scale dataset with 18,381 nodes, involving high-dimensional sports performance metrics. MovieLens is a large-scale dataset, consisting of 1,000,209 nodes, and presents challenges related to sparsity and efficiency in large data environments.

These datasets have been used in previous work on multi-criteria decision-making and skyline query processing (Gulzar et al., 2021; Mohamud et al., 2023). Their inclusion allows for direct performance comparisons in terms of accuracy, query response time, and memory usage. Finally, the real-world applicability of these datasets, spanning customer analysis, sports analytics, and recommendation systems, ensures the practical relevance and impact of the research in real-life scenarios.

3.3.2 Summary of Real-World Datasets

This study evaluates the proposed framework using three real-world datasets: CoIL 2000, NBA Stats, and MovieLens, which were selected for their diversity in domain, data volume, and feature complexity. These datasets are commonly referenced in prior literature related to multi-criteria decision-making, incomplete data handling, and recommendation systems. To simulate real-world scenarios involving incomplete information, 20% attribute-level incompleteness was introduced to each dataset, either through random removal or by building on pre-existing missing values.

For CoIL 2000, which contains 5,822 customer records, the data is fully complete and includes 86 attributes divided into sociodemographic data (attributes 1–43) and product ownership (attributes 44–86). The sociodemographic data is derived from zip code information. The dataset was provided in 2000 by Sentient Machine Research, a Dutch data mining company, and is rooted in a real-world business problem. In alignment with one previous study on skyline query processing in incomplete data, 20% of attribute values were removed at random to introduce controlled incompleteness.

The NBA Stats dataset aggregates individual player performance from the 1946 to 2005 basketball regular seasons, incorporating historical data from the BAA, NBA, and ABA. It includes 18,381 entries and 17 features per record. Each entry represents a player's average performance per game in a season, and players may appear multiple times if they played for multiple teams. Features include standard demographics and performance statistics. The features are games, games started, minutes played, field goals, field goal attempts, field goal percentage, free throws, free throw attempts, free throw percentage, offensive rebounds, defensive rebounds, total rebounds, assists, steals, personal fouls, points, and triple-doubles. This dataset is commonly referred to as "NBA Stats" in the literature. While the dataset contains some pre-existing missing values, additional incompleteness was simulated to reach 20% total missing values, in accordance with common practices in the literature.

The MovieLens dataset includes 1,000,209 ratings from 6,040 users for approximately 3,900 movies, released in 2003. Each user contributed at least 20 ratings, made on a 5-star scale. The dataset is complete in its raw form, with each row recording a user ID, movie ID, rating, and timestamp. To simulate real-world sparsity and missing preferences, 20% of the attribute values were randomly removed, following the approach taken in many prior works.

Snippets from the CoIL 2000, NBA Stats, and MovieLens datasets are provided in Appendix III, Appendix IV, and Appendix V for clarity and reproducibility.

The table below summarizes key characteristics of each dataset and details the approach taken to introduce and define incompleteness. The focus is on attribute-level incomplete graphs, where node features may be partially missing while maintaining structural relationships through similarity-based edges.

Table 3.1 Summary of Real-World Datasets and Incompleteness Characteristics

Dataset	No. of Nodes	No. of Columns	Original Completeness	Incompleteness Introduced	Incomplete Graph Type	Graph Representation
CoIL 2000	5,822	86	Fully complete	20% attribute values removed randomly	Attribute-level incompleteness	Nodes: customers; Edges: feature similarity
NBA Stats	18,381	17	Partially incomplete	Removed attribute values to reach 20% total missing values	Attribute-level incompleteness	Nodes: players; Edges: statistical similarity
MovieLens	1,000,209	4	Fully complete	20% attribute values removed randomly	Attribute-level incompleteness	Nodes: users; Edges: preference similarity

This structured overview clarifies the real-world grounding and controls experimental manipulation of each dataset. By focusing on attribute-level missingness, the study closely mirrors practical challenges faced in real-life decision support and recommender systems, where complete data is rarely available. The datasets collectively allow the framework to be tested across varied contexts in terms of data scale, complexity, and application domain.

3.3.3 Synthetic Data Generation

The use of synthetic data in this study is both justified and necessary for several reasons. First, synthetic data generation allows for complete control over key variables such as dataset size, attribute distributions, dimensionality, and the proportion of missing values. This control is critical for systematically evaluating the performance of the

proposed framework across a range of scenarios, including edge cases that are difficult to isolate in real-world datasets. Synthetic datasets are commonly used in skyline research for benchmarking under controlled settings (Bharuka and Kumar, 2013b; He & Han, 2022). Previous studies have relied on synthetic data to simulate structured environments with varying levels of data incompleteness, enabling consistent benchmarking and fair comparisons among competing algorithms.

In this research, the synthetic dataset is designed to closely simulate real-world data conditions by incorporating features such as price, customer rating, availability, and shipping time—attributes that are commonly found in domains like e-commerce, logistics, and decision support systems. Incompleteness is introduced at controlled levels (10%, 50%, and 90%) to reflect realistic scenarios of missing data due to user omission, sensor failure, integration errors, or incomplete transactions. The dataset also includes both numerical and categorical attributes, ensuring it mimics the heterogeneous nature of real applications. This structured yet flexible approach to synthetic data generation allows for rigorous testing of scalability, accuracy, and robustness, and ensures that the experimental outcomes are grounded in conditions commonly found in real-world datasets.

3.3.4 Summary of Synthetic Datasets

To thoroughly evaluate the robustness of the proposed framework under varying levels of attribute-level incompleteness, four synthetic datasets were generated, each simulating a real-world e-commerce platform. Synthetic datasets are widely used in previous literature on skyline query processing for incomplete graphs, as they offer controlled environments to isolate the effects of missing data on algorithmic performance.

Each dataset contains 5,000 nodes, where each node represents a product, and six attributes are associated with each product: Product ID, Price, Rating, Availability, Shipping Time, and Category. The graph is constructed by connecting products that share the same category, forming 1,249,533 edges, which yields an edge density of approximately 0.1 (9.998%) out of a possible 12,497,500 node-to-node connections.

The first dataset introduces targeted incompleteness to simulate a heterogeneous missing data pattern. Specifically, availability has 20% missing values, price and shipping time each have 10% missing values, and rating has 5% missing values. This dataset was used in preliminary work.

Given the dataset contains 30,000 data cells (6 columns × 5,000 rows), this results in 1,000 missing cells in availability, 500 each from price and shipping time, and 250 from rating. This totals 2,250 missing values, or 7.5% incompleteness across the dataset.

For the remaining three datasets, incompleteness is introduced uniformly at random across all attributes at fixed levels of 10%, 50%, and 90%, respectively. This progressive degradation of data quality allows for rigorous robustness testing of framework under scalable form of missingness.

A sample from the 10% incompleteness synthetic dataset is provided in Appendix II for clarity and reproducibility.

Table 3.2 Summary of Synthetic E-Commerce Datasets and Incompleteness Characteristics

Dataset	**No. of Nodes**	**No. of Columns**	**No. of Edges**	**Max Possible Edges**	**Edge Density**	**Attribute Missingness**
Synthetic-7.5%	5,000	6	1,249,533	12,497,500	~0.1 (9.998%)	Availability: 20%; Price & Shipping Time: 10% each; Rating: 5%
Synthetic-10%	5,000	6	1,249,533	12,497,500	~0.1 (9.998%)	10%
Synthetic-50%	5,000	6	1,249,533	12,497,500	~0.1 (9.998%)	50%

Synthetic-90%	5,000	6	1,249,533	12,497,500	~0.1 (9.998%)	90%

These datasets provide a comprehensive testing environment for benchmarking the robustness of skyline algorithms and proposed framework. By varying the degree of incompleteness while holding the graph topology constant, the study ensures that performance differences across methods are directly attributable to their handling of missing data. This controlled design is essential for deriving reliable conclusions about algorithmic behavior in incomplete graph settings.

3.3.5 Preprocessing

Data preprocessing is a critical step in preparing both synthetic and real-world datasets for effective skyline query computation using machine learning models, particularly Graph Convolutional Network (GCN). The preprocessing pipeline in this study consists of several key stages: data normalization, missing value handling, graph construction, and dataset partitioning. First, numerical attributes such as price, rating, availability, and shipping time are normalized to a common scale (typically between 0 and 1) to ensure that no single feature dominates during skyline computation or model training. Normalization also facilitates the learning process in GCN by maintaining consistent feature magnitudes across nodes.

Handling missing data is particularly important in this research due to the focus on incomplete graphs. The framework is designed to operate on datasets with attribute-level incompleteness. During skyline ground truth labelling with ISkyline, missing values are handled by skipping comparisons on null entries, preserving skyline semantics without requiring imputation. For GCN training, missing feature values are left as-is and implicitly learned from neighboring nodes via the message-passing mechanism (Kipf & Welling, 2017). This allows the model to infer missing information contextually, eliminating the need for explicit statistical or mean-based imputation.

Following data cleaning and imputation, each dataset is converted into a graph format where individual data records become nodes, and edges are formed based on

attribute similarity or shared categorical properties (e.g., a product category). This transformation enables skyline queries to operate over graph-structured data, which is essential for applying GCN.

After graph construction, the dataset is split into training, validation, and testing subsets using an 80:10:10 ratio. The split is stratified based on skyline vs. non-skyline labels to ensure balanced class representation across all subsets. For the GCN, training and validation subsets are used for model optimization, while the testing subset evaluates generalization performance on unseen data.

This comprehensive preprocessing approach ensures that the data fed into the framework is clean, structured, and well-suited for the learning models employed, allowing for robust skyline query computation across varying levels of data completeness and scale.

3.4 SELECTION AND IMPLEMENTATION OF MACHINE LEARNING MODELS

3.4.1 Baseline Models

To evaluate the effectiveness of the proposed machine learning-based framework, traditional skyline query algorithms were selected as baseline models. Specifically, ISkyline, SIDS, and OIS were chosen due to their widespread use in existing research on skyline query processing, particularly in the context of incomplete datasets and large-scale graphs (Bharuka & Kumar, 2013a). These algorithms represent well-established and diverse approaches within the field, making them suitable candidates for performance benchmarking (Gulzar et al., 2019).

ISkyline is commonly referenced in prior literature as a method optimized for handling incomplete data. It utilizes bitmap representations, shadow skylines, and virtual points to reduce the number of dominance comparisons required. Its approach to managing missing values through bucket-based partitioning and shadow skyline approximation makes it a relevant baseline for evaluating improvements in robustness and processing efficiency.

SIDS (Sort-based Incomplete Data Skyline) offers another commonly used approach in the field, particularly for datasets with partially missing attributes (Bharuka

& Kumar, 2013a). It applies a round-robin sorting mechanism that incrementally prunes dominated tuples, which helps reduce computation in sequential access scenarios. Though efficient for static datasets, its limitations with real-time adaptability and high-dimensional incompleteness make it a strong point of comparison for newer dynamic models.

OIS (Optimized Incomplete Skyline) is included as a baseline for its strength in minimizing I/O overhead in large-scale skyline queries (Gulzar et al., 2019). It is frequently used in scenarios where disk-based processing is required, and computational resources are constrained. Although OIS is not inherently designed to manage missing data, its efficiency in large datasets makes it a valuable benchmark for evaluating scalability and query response time.

By selecting ISkyline, SIDS, and OIS, this study aligns with the precedent set by existing literature and ensures that the performance of the proposed framework is compared against well-recognized, foundational methods. This comparison provides meaningful insights into how modern, learning-based solutions improve upon traditional skyline processing, particularly in incomplete and dynamic graph environments.

3.4.2 Graph Convolutional Network (GCN) Model

To capture dominance relationships in large-scale and incomplete graphs, this research proposes the development of a Graph Convolutional Network (GCN)-based framework. GCN are particularly well-suited for this task due to their inherent capability to model graph-structured data (Kipf & Welling, 2017). Skyline queries often involve multi-dimensional datasets, where dominance relationships between data points can naturally be represented as a graph. GCN not only model the attributes of individual nodes but also capture the complex interdependencies and dominance relationships among them (Kipf & Welling, 2017), making them an ideal choice for this context.

One of the key advantages of GCN is their ability to handle incomplete data effectively (Kipf & Welling, 2017). By leveraging information from neighboring nodes, GCN can infer or fill in missing attributes, allowing for more accurate and robust data analysis. Unlike traditional imputation techniques, GCN dynamically learn which

attributes and relationships are most important, resulting in more context-aware imputations.

Scalability and efficiency are also major strengths of GCN (Kipf & Welling, 2017). Their design enables parallel processing and makes them inherently scalable to large datasets. During training and inference, GCN focus on local neighborhoods rather than performing exhaustive pairwise comparisons, which significantly reduces computational overhead. Additionally, GCN are highly adaptable to dynamic environments (Hamilton et al., 2017). Their message-passing mechanisms allow for incremental updates, meaning that changes such as data insertions or deletions can be incorporated without reprocessing the entire dataset which is an essential feature for real-time applications of skyline queries.

Furthermore, GCN integrate well into broader machine learning pipelines, supporting custom architectures like attention mechanisms or hierarchical structures that can be tailored to specific requirements of skyline query processing. Empirical evidence from recent studies also highlights the superiority of GCN over conventional algorithms, particularly in terms of accuracy, query response time, and memory efficiency for graph-related tasks.

Overall, the use of GCN aligns closely with the research goals of improving scalability, adaptability, and efficiency in skyline queries over large-scale and incomplete graphs. By effectively leveraging both local and global graph structures, GCN offer a modern and innovative solution to overcome the limitations of traditional methods.

3.4.3 Frameworks

In the development of Graph Convolutional Network (GCN) for skyline queries, frameworks like PyTorch Geometric (PyG) and Deep Graph Library (DGL) are invaluable due to their efficiency, flexibility, and extensive support for graph-based computations. PyG, built on PyTorch, provides an intuitive and modular API for constructing and training GCNs, offering pre-implemented layers (e.g., GNN, GAT, GraphSAGE) and utilities for handling large-scale graphs with sparse connections. Its ability to seamlessly integrate with the PyTorch ecosystem enables easy customization

of models, experimentation with advanced architectures, and utilization of GPU acceleration for scaling computations. Similarly, DGL, another powerful framework, is designed to handle dynamic and complex graph structures efficiently. It supports heterogeneous graphs, enabling the representation of multi-relational data, which is especially useful for skyline queries involving diverse attributes (Yang et al., 2021). DGL also provides optimization techniques like mini-batch training and graph sampling, which are crucial for processing large-scale graphs in skyline computations. Both frameworks simplify the implementation of message-passing mechanisms, feature extraction, and embedding generation, ensuring scalability and adaptability to dynamic datasets. Leveraging these frameworks in GCN development not only accelerates research but also enhances the accuracy and efficiency of skyline query models in handling incomplete or large datasets.

3.4.4 Architecture

The architecture for leveraging Graph Convolutional Network (GCN) in skyline queries involves representing the multi-dimensional dataset as a graph, where each data point becomes a node, and edges signify relationships between nodes based on dominance criteria. Node embeddings are used to encode the attributes of each data point, such as cost, performance, or distance, into a vectorized form suitable for computation. Additionally, edge attributes can be incorporated to represent the specific nature of the relationship between connected nodes, such as the degree of dominance or shared dimensions. The GCN processes this graph structure using message-passing techniques, where information is iteratively exchanged between neighboring nodes along the edges. During each message-passing iteration, nodes aggregate information from their neighbors and update their embeddings, effectively capturing both local and global graph-level features. For example, a node can learn its relative position in the dominance hierarchy by propagating and receiving feature vectors from its neighbors. The final embeddings generated after multiple message-passing layers can be used to classify nodes as skyline or non-skyline points. This architecture can further benefit from advanced techniques like attention mechanisms, which weigh the importance of each neighbor, and hierarchical pooling, which summarizes subgraph features to

improve scalability and accuracy. Such a design ensures that the model effectively handles complex relationships in large-scale and incomplete datasets, providing robust and accurate skyline computations.

3.5 DYNAMIC ADAPTABILITY

GCN offer a natural capacity for adaptability in graph-structured data by generalizing learning across nodes and edges, even when the underlying data exhibits incomplete or evolving patterns.

One of the primary advantages of GCN in dynamic environments is their ability to leverage structural and feature-based information jointly (Kipf & Welling, 2017). When the dataset is updated, such as by introducing new nodes, modifying edge weights, or encountering previously missing attribute values, the model can continue to infer meaningful relationships without needing complete retraining from scratch. GCN learn localized patterns across node neighborhoods (Kipf & Welling, 2017), which makes them particularly effective in handling changes or partial updates in the data while preserving scalability.

In the context of skyline query processing, adaptability is also achieved by the model's ability to generalize dominance patterns observed in training to new or previously unseen data points. Since the GCN is trained on a dominance graph where edges reflect the outcome of the skyline algorithm dominance function, the model can capture relational dependencies and skyline-relevant patterns that persist even if node attributes are partially incomplete or updated dynamically (Khalefa et al., 2008: Kipf & Welling, 2017). This allows the system to remain robust and responsive under varying data conditions without requiring full recomputation of the skyline each time the data changes.

Furthermore, the modular architecture of the GCN enables it to be retrained incrementally on updated datasets. While not implementing full online learning, this retraining can be done efficiently on newer data snapshots, using the pre-trained model as a base. This strikes a balance between performance and responsiveness, making it a viable alternative to traditional online or reinforcement learning approaches.

In summary, GCN provides dynamic adaptability by learning generalizable patterns over graph structures, handling incomplete or evolving data, and supporting efficient retraining on updates. This positions GCN as a flexible and powerful mechanism for real-time skyline query processing in dynamic graph environments.

3.6 STEP-BY-STEP PROCESS FROM RAW DATA TO SKYLINE VISUALIZATION

This subsection presents a step-by-step process that transforms raw data into a skyline visualization using a subset of NBA player statistics as an example.

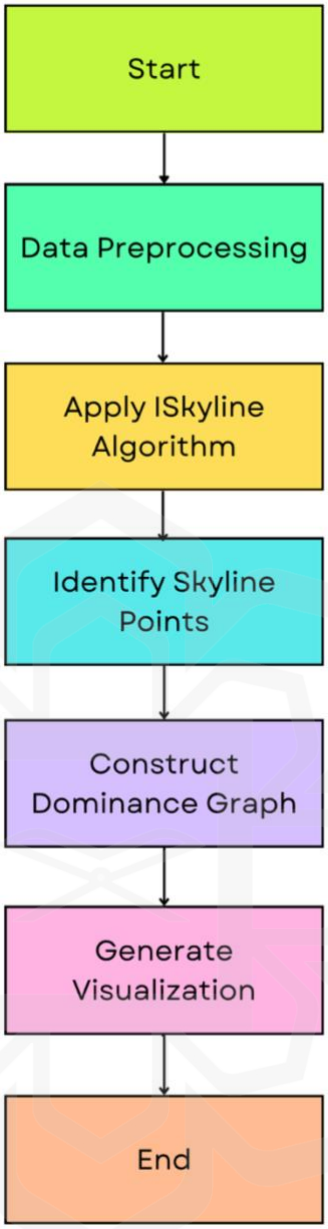


Figure 3.4 Process from Raw Data to Skyline Visualization

Figure 3.4 illustrates the step-by-step process from raw data acquisition through preprocessing, ISkyline algorithm application, skyline point identification, and dominance graph construction, to the final visualization of skyline results.

	B	C	D	E
1	GS	MP	FG	FGA
2	80	3277	978	2173
3	79	3361	875	1823
4	72	3103	815	1822
5	80	3384	746	1668
6	81	3089	751	1564
7	80			
8	79	3084	689	
9		2892	699	1413
10	80	3130		1516

Figure 3.5 Subset of NBA Player Statistics

The dataset shown in Figure 3.5 includes attributes such as Games Started (GS), Minutes Played (MP), Field Goals (FG), and Field Goal Attempts (FGA). For demonstration, a small sample of players is selected to illustrate how skyline queries are processed to identify Pareto-optimal results.

The process begins with data preprocessing, where missing values in attributes like FGA are handled by skipping dominance comparisons involving those null entries, following the ISkyline method. Attributes such as MP and FG are normalized to a standard scale to ensure fair comparisons. A dominance graph is then constructed where each player is represented as a node, and edges are created based on dominance relationships. For example, a player with more field goals and minutes played will dominate another player.

Next, the ISkyline algorithm is applied to identify skyline points, which are players who are not dominated by any other in the dataset. These skyline points represent the most performing players, with high field goals and high minutes played. Non-skyline players are those who are outperformed by at least one skyline player across the considered dimensions.

The dominance graph constructed from ISkyline is then used to train a Graph Neural Network (GNN), specifically a Graph Convolutional Network (GCN), where the model learns to classify players as skyline or non-skyline based on their features and

dominance relationships. This GCN model helps infer skyline membership even when some attributes are incomplete by learning from the structure of the graph.

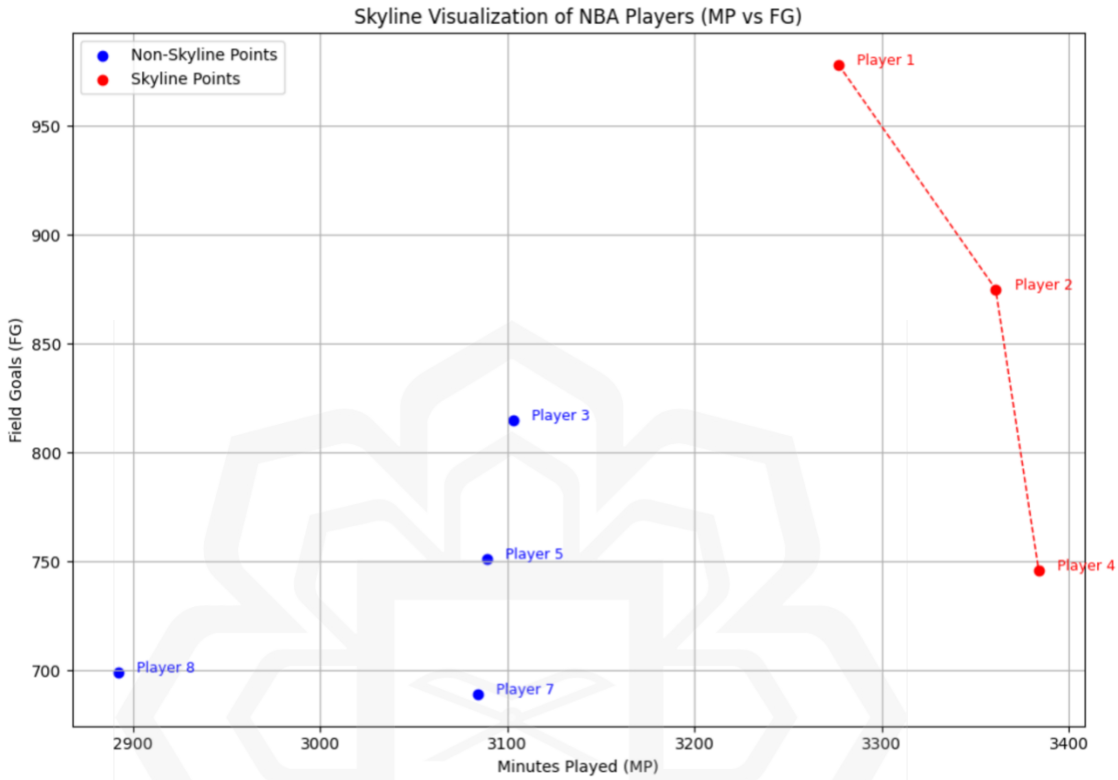


Figure 3.6 Skyline Visualization of NBA Players (MP vs FG)

Finally, a skyline visualization is generated, plotting Field Goals (FG) against Minutes Played (MP), where skyline points form a Pareto frontier representing the optimal trade-offs. Figure 3.6 clearly shows which players are performing (skyline) and which are dominated (non-skyline). The step-by-step process, from raw data to skyline visualization, illustrates the practical application of the proposed hybrid method in handling multi-dimensional and incomplete data scenarios.

3.7 CHAPTER SUMMARY

This study employs a design science research methodology based on Hevner's (2004) framework, incorporating the Relevance Cycle (connecting research to its context), Rigor Cycle (leveraging scientific knowledge), and Design Cycle (developing and evaluating artifacts). To address scalability, incomplete data handling, and dynamic adaptability in large-scale graphs, real-world datasets (e.g., CoIL 2000, NBA Stats, MovieLens) and synthetic datasets with varying levels of incompleteness (10%, 50%, 90%) are used. Data preparation includes normalization, handling missing values through imputation or Graph Convolutional Network (GCN)-based methods, and splitting datasets into training, validation, and test sets. Traditional skyline algorithms (e.g., ISkyline, SIDS, OIS) serve as baselines, while a GCN-based framework, built using tools like PyTorch Geometric, captures dominance relationships in large, incomplete graphs using node embeddings, edge attributes, and message-passing techniques.

CHAPTER FOUR

EXPERIMENTAL RESEARCH DESIGN

4.1 INTRODUCTION

The preceding chapter outlined the research methodology employed in this study. Building upon that foundation, this chapter focuses on the experimental research design, which is pivotal for structuring the data collection and analysis processes.

This step aims to organize experiments to achieve the research objectives and validate the hypotheses. Key procedures that require detailed explanation include evaluation and benchmarking, experiment settings, benchmark against traditional methods, visualization of results, potential findings and preliminary outcomes, and experimental limitations.

4.2 EVALUATION AND BENCHMARKING

4.2.1 Definitions of Scalability and Robustness

In the context of skyline query processing and GCN-based classification, scalability refers to the system's ability to maintain performance as the dataset size increases (Dean & Ghemawat, 2008). It is measured primarily by tracking changes in computation time and memory usage as the number of records (nodes) or features (dimensions) grows. A scalable method should exhibit sub-linear or controlled growth in runtime when applied to increasingly large datasets (Henzinger et al., 2020).

Robustness, on the other hand, refers to the framework's ability to maintain classification accuracy and stability in the presence of data incompleteness, noise, or random perturbations (Zhu & Wu, 2004). We assess robustness by evaluating model performance on datasets with artificially introduced missing values and by testing sensitivity to different levels of data sparsity. A robust model should show minimal

degradation in F1-score, precision, or recall under such adverse conditions (García et al., 2015).

These two characteristics are critical for real-world deployment, where datasets are often incomplete, high-dimensional, and subject to change.

4.2.2 Measuring Accuracy, Precision, Recall, F1-Score and AUC-ROC

To evaluate the effectiveness of the proposed GNN + ISkyline framework and baseline models in classifying skyline and non-skyline points, five widely used performance metrics are employed: accuracy, precision, recall, F1-score, and AUC-ROC. These metrics provide a comprehensive view of the model's classification performance, especially in scenarios involving class imbalance, which is common in skyline query tasks where skyline points typically represent a minority class (Afifi et al., 2024).

Accuracy is the ratio of correctly predicted instances to the total number of predictions made. While it provides a general measure of correctness, accuracy alone may be misleading in imbalanced datasets. It is computed as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

Precision measures the proportion of correctly predicted skyline points among all points classified as skyline by the model. It reflects the model's ability to avoid false positives and is given by:

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

Recall (also known as sensitivity or true positive rate) indicates the proportion of actual skyline points that were correctly identified by the model. A high recall shows that the model effectively captures most of the relevant skyline instances. The formula for recall is:

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

F1-score is the harmonic mean of precision and recall. It provides a single score that balances both false positives and false negatives, which is particularly important in cases of uneven class distribution. F1-score is defined as:

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (6)$$

AUC-ROC (Area Under the Receiver Operating Characteristic Curve) evaluates the model's ability to distinguish between classes across all classification thresholds. It plots the true positive rate against the false positive rate and summarizes the model's discrimination capability regardless of the threshold. An AUC-ROC of 1.0 indicates perfect classification, while 0.5 suggests random guessing.

Together, these metrics offer a balanced and robust evaluation framework for assessing the predictive performance of skyline classification models. These metrics are widely used in evaluating binary classifiers, particularly under class imbalance (Afifi et al., 2024). They are computed on the test dataset and are particularly important when validating models trained on datasets with varying levels of incompleteness. High scores across these metrics indicate not only the correctness of predictions but also the reliability of the model in detecting minority skyline points without overfitting or bias.

4.2.3 Measuring Query Response Time

Query response time is a critical performance metric for evaluating the efficiency of skyline query processing methods, particularly when dealing with large and incomplete datasets. In this study, query response time is defined as the total time taken from the initiation of the query processing task to the point at which the final results are generated.

For traditional skyline algorithms such as ISkyline, SIDS, and OIS, the response time is measured from the moment the query is issued, starting with data loading, through to the point where skyline points are computed and returned. This includes all internal computations such as dominance checks, sorting, and pruning strategies, which contribute directly to the overall processing latency.

For machine learning-based methods, including the proposed GNN + ISkyline framework, the query response time encompasses the skyline query processing and the entire machine learning pipeline. Specifically, the timer begins immediately before loading the preprocesses data and continues through the stages of model inference and evaluation. This includes converting the input into model-compatible formats (e.g., tensors), applying the trained model to make predictions, and computing evaluation

metrics such as accuracy, F1-score, and AUC-ROC. By capturing the full scope of operations involved in producing the final skyline classification, this measurement reflects the practical performance users would experience in a real-time or near-real-time system.

Measuring end-to-end query latency, including dominance computation and model inference, follows common benchmarking practices in skyline and real-time systems research (He & Han, 2022). All timing measurements are performed under consistent conditions on the same hardware platform to ensure comparability across methods. This allows a fair assessment of both algorithmic complexity and computational efficiency in dynamic, incomplete data environments.

4.2.4 Measuring Memory Usage

Memory usage is an essential metric for evaluating the computational efficiency of skyline query processing methods, especially when applied to large-scale and incomplete datasets. It provides insights into how resource-intensive each method is, which is particularly relevant for deployment in memory-constrained environments. In this study, memory usage is measured as the peak memory usage observed during the complete execution cycle of each method.

In the case of machine learning-based methods, including the proposed GNN + ISkyline framework, memory usage is measured starting immediately before loading the input data and continues through the stages of data transformation, model inference, and evaluation. This includes the memory consumed by tensor creation, model loading, forward propagation, and the calculation of evaluation metrics like accuracy, F1-score, and AUC-ROC. Measuring memory usage across the full pipeline ensures a fair comparison between learning-based and algorithmic methods, especially since neural models may involve additional memory overhead for model weights, batch operations, and GPU or CPU memory buffers.

All measurements are taken in a consistent runtime environment on the same hardware to ensure comparability. The peak memory usage is recorded using system-level memory profiling tools, providing an accurate representation of each method's resource demand under realistic conditions.

4.2.5 Integration of Performance Metrics

Once query response time and memory usage have been measured, these results should be analyzed in relation to other key performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC to provide a comprehensive evaluation of the proposed method. This enables a meaningful trade-off analysis between predictive performance and computational efficiency.

For instance, it's important to highlight the trade-off between accuracy and query response time. In some scenarios, achieving higher accuracy may come at the cost of increased computational complexity, leading to longer response times. Similarly, as the dataset size or graph density increases, memory usage tends to rise. This relationship should be compared with traditional methods to assess whether the proposed approach maintains scalability more effectively or incurs similar limitations.

To better communicate these trade-offs, visualization tools can be employed. Multi-axis plots are useful for illustrating the relationship between accuracy and other metrics like query response time or memory usage. Additionally, summarizing accuracy, precision, recall, F1-score, AUC-ROC, query response time, and memory usage in comparative tables allows for a more holistic view, helping stakeholders understand the balance between efficiency and effectiveness in different application contexts.

4.3 EXPERIMENT SETTINGS

4.3.1 Baseline Configuration Details

In order to fairly evaluate the performance of the proposed GNN + ISkyline framework, the proposed framework is compared against three key baseline methods, ISkyline, SIDS, and OIS. Each method was selected due to its distinct approach to skyline computation under incomplete or high-dimensional data conditions. The configurations and parameters for each were standardized and tuned where appropriate to ensure consistency and fairness in the comparison.

ISkyline serves as the primary dominance-based baseline. It computes skyline points over datasets with missing values by evaluating partial dominance relationships

and eliminating tuples that are fully dominated within observed attribute subsets. In this study, ISkyline is implemented with its default dominance logic, and when attribute values are missing, comparisons are conducted only on overlapping attributes. Tie-breaking is resolved using tuple ordering by ID to maintain consistency.

SIDS provides another baseline specifically designed for data with null values. SIDS determines skyline membership by considering only the attributes shared between two tuples during pairwise dominance checks. In these experiments, SIDS is configured to exclude tuples with more than 50% missing attributes to ensure meaningful dominance comparisons. It uses standard logical filtering and applies skyline rules only on the observed dimensions, thereby allowing a more tolerant skyline set in the presence of incompleteness.

OIS represents an efficiency-oriented enhancement of skyline computation. OIS incrementally refines candidate skylines using iterative dominance checks and early pruning strategies. These experiments adopt the original algorithmic structure of OIS and apply it uniformly across all synthetic and real-world datasets. The iteration threshold is tuned based on dataset size, and the same normalization and missing value handling procedures are applied as with the proposed method, ensuring comparability in both runtime and performance assessments.

All three baseline methods are evaluated under the same experimental setup, including consistent data normalization, 80/10/10 train-validation-test splitting, and the use of identical datasets. Their inclusion offers a comprehensive performance benchmark covering traditional, incomplete-aware, and optimized skyline computation strategies.

4.3.2 Scalability

To assess the scalability of the proposed framework, three real-world datasets of increasing size and complexity were selected: CoIL 2000, NBA Stats, and MovieLens. These datasets represent small, medium, and large-scale graph scenarios, respectively, and were chosen to evaluate how the framework performs under varying data volumes.

The CoIL 2000 dataset, comprising 5,822 nodes, represents a small to medium-sized dataset typically used in customer relationship management scenarios. This dataset was ideal for baseline testing, offering insights into how the framework performs in controlled, moderately sparse graph environments with categorical and numerical attributes.

The NBA Stats dataset, consisting of 18,381 nodes, was used to test the model on a more complex and denser dataset that involves a wide range of performance metrics for professional basketball players. This dataset simulates scenarios in sports analytics where real-time decision-making is essential.

The MovieLens dataset, with over 1,000,209 nodes, served as a large-scale benchmark to assess the framework's ability to handle high-volume, sparse, and user-centric data. As one of the most widely used datasets in recommendation system research, MovieLens presented a significant challenge due to its scale and the sparsity of user-item interaction data.

Each dataset was converted into a graph format by modeling individual records as nodes and establishing edges based on similarity or shared attribute values (Hamilton et al., 2017). This experimental setup allows for evaluating the proposed method's performance under realistic scaling conditions and across diverse application domains. The evaluation focused on measuring accuracy, F1-score, query response time, and memory usage under increasing graph sizes.

For each dataset, skyline query processing was conducted using the proposed framework and baseline methods. Key performance metrics such as accuracy, F1-score, query response time, and memory usage were recorded and plotted across datasets to show how each method scales with the number of nodes. Visualization was done using line and bar charts to compare performance trends. The horizontal axis in each graph represents dataset size (number of nodes), while the vertical axis reflects the specific performance metric (e.g., accuracy in percentage, F1-score in percentage, response time in seconds, memory usage in MB).

In the query response time visualization, the performance of the proposed method is plotted alongside traditional algorithms like ISkyline, SIDS, and OIS. The curve demonstrates how query execution time increases with the dataset size, highlighting the differences in scalability across methods. For memory usage, similar charts were constructed to show peak memory consumption as dataset complexity

grows. These visualizations provide a clear, comparative view of how the framework manages computational resources across small (CoIL 2000), medium (NBA Stats), and large-scale (MovieLens) datasets.

To support interpretability, each chart is accompanied by descriptive labels, legends, and annotations where necessary. These visual tools help illustrate the efficiency and scalability of the framework in handling real-world, graph-structured data under diverse conditions. The plots were generated using Python libraries such as Matplotlib and Seaborn for consistency and clarity.

4.3.3 Robustness

To evaluate the robustness of the proposed framework, experiments were conducted using synthetic datasets with varying levels of data incompleteness. Specifically, missing values were introduced in 10%, 50%, and 90% of the dataset attributes to simulate different levels of real-world data degradation. These levels represent low, moderate, and high incompleteness scenarios, respectively. The synthetic datasets were designed with predefined dominance relationships, allowing controlled testing of the framework's ability to maintain skyline query performance despite missing information.

During the experiment, missing values were randomly distributed across key numerical attributes such as price, rating, availability, and shipping time. The same base synthetic dataset was used in each case, with only the percentage of missing data altered. This ensured consistency in testing and allowed for accurate comparisons across the three levels of incompleteness. Skyline queries were then executed using both the proposed framework and traditional baseline methods, and metrics such as accuracy, F1-score, query response time, and memory usage were recorded.

The robustness evaluation was particularly focused on the framework's ability to impute missing values accurately and maintain effective skyline filtering under increasingly sparse data conditions. By incrementally increasing the level of missingness, the experiment tested the limits of the framework's learning capacity and its reliance on the graph structure for relational inference. These tests were critical to

understanding how the system performs when confronted with real-world challenges like sensor failure, user omission, or incomplete data integration.

To visualize the results of the robustness tests, performance metrics were plotted against the percentage of incompleteness (10%, 50%, and 90%) (Bharuka and Kumar, 2013a). Line graphs were created to show how each method's accuracy changes as more data becomes missing, providing insight into the resilience of the algorithms. Separate charts were also developed for F1-score, query response time, and memory usage.

In the accuracy visualization, the vertical axis represents the percentage of correctly identified skyline points, while the horizontal axis denotes the level of data incompleteness. A clear trend line helps compare the degradation in performance between methods, highlighting which approaches are more tolerant of missing data. For F1-score, query response time, and memory usage, similar plots show how classification effectiveness and computational overhead shift under different levels of missingness.

These visualizations help identify the threshold at which traditional algorithms begin to fail and the point where the proposed framework maintains stable performance. Each figure is clearly labelled, and legends differentiate between the methods tested. Python libraries such as Seaborn and Matplotlib were used to ensure a clear and consistent presentation of the results. Together, these plots provide an intuitive understanding of the system’s robustness and its capacity to operate effectively in incomplete data environments.

4.4 MODEL REFINEMENT PROCESS

4.4.1 Overfitting Countermeasures

To ensure that the proposed GCN + ISkyline model generalizes well and does not overfit to the training data, several proactive strategies were incorporated during the experimental setup. Overfitting is a common challenge in deep learning models, particularly in graph-based neural networks where model complexity and node connectivity can lead to memorization rather than learning general patterns (Zhang & Chen, 2018). Recognizing this risk, both regularization techniques and validation strategies were applied to safeguard the robustness of the results.

One of the primary measures used to prevent overfitting was the implementation of dropout regularization. In the GCN architecture, dropout layers with a probability of 0.5 were applied after each graph convolutional layer. This technique randomly deactivates a subset of neurons during training, which encourages the model to learn redundant representations and reduces dependency on specific paths in the graph structure (Srivastava et al., 2014). Additionally, weight decay was applied as part of the Adam optimizer configuration. This acts as L2 regularization, penalizing large weights during training and further promoting model generalization (Ng, 2004).

Beyond regularization, cross-validation techniques were employed to assess the stability and generality of the model's performance. Specifically, k-fold cross-validation with k = 5 was used during preliminary hyperparameter tuning. This approach ensured that the performance metrics reported were not biased by a single train-test split. Each fold involved training the model on 80% of the data and validating and testing it on the remaining 20%, with results averaged across all folds to produce a more reliable estimate of performance.

Furthermore, the dataset was shuffled and split randomly in each training iteration to prevent the model from memorizing patterns based on data order. Early stopping was also considered in initial trials to monitor validation loss across epochs and halt training if no improvement was observed within a defined patience threshold, although final training runs opted for a fixed number of 200 epochs (Prechelt, 1998).

In combination, these cross-validation and regularization techniques served as robust countermeasures against overfitting, ensuring that the reported metrics reflect genuine predictive capability rather than model overfitting to the training set.

To further mitigate the risk of overfitting, especially given the imbalanced nature of skyline versus non-skyline data points, future iterations of the framework could benefit from incorporating more rigorous regularization strategies and model validation techniques. For instance, cross-validation could be applied in conjunction with early stopping to prevent the model from learning noise or redundant patterns specific to the training set.

Moreover, feature ablation studies and dropout rate optimization could provide further insights into which attributes most influence the skyline classification task and whether the model is overly reliant on any single input. These extensions would

strengthen the empirical foundation of the framework and ensure its robustness across broader and more diverse datasets.

4.4.2 Hyperparameter Tuning Strategy

In the training of the GNN + ISkyline model, grid search was employed as the chosen strategy for hyperparameter tuning. This decision was made based on its simplicity, transparency, and effectiveness in systematically exploring a predefined parameter space. Grid search exhaustively evaluates all possible combinations of specified hyperparameter values, which ensures that the best-performing configuration within the defined range is identified. Given the moderate number of tunable parameters, such as the learning rate, number of hidden units, dropout rate, and weight decay, the computational cost remained manageable and did not warrant the complexity of more advanced techniques like Bayesian optimization or random search.

Moreover, grid search provides reproducibility and interpretability, which are essential during early experimental phases when understanding the effects of each hyperparameter on model performance is crucial. For instance, tuning was conducted over learning rates [0.001, 0.005, 0.01], hidden dimensions [16, 32, 64], and dropout rates [0.3, 0.5, 0.7], and results were evaluated based on cross-validation accuracy and F1-score. These exhaustive comparisons allowed for clearer insights into the sensitivity of the model to different configurations and helped ensure that the selected settings were not arbitrary.

While grid search may not scale efficiently to extremely large hyperparameter spaces, it was a suitable and justified approach in this context due to the constrained search space and the interpretability requirements of the experiment.

4.5 SETUP AND CONFIGURATION

All experiments and model implementations in this research were conducted using a MacBook Air powered by the Apple M2 chip, featuring an 8-core CPU, 8-core GPU, and a 16-core Neural Engine. The system was equipped with 8 GB of unified memory

and a 256 GB SSD, running on macOS Sequoia. This hardware configuration, while not classified as a high-end workstation, offers a well-balanced performance environment optimized for energy-efficient machine learning workflows, especially those involving lightweight graph-based models and neural network architectures.

The proposed framework was implemented using Python 3.11, with core dependencies including PyTorch (via the MPS backend for Apple Silicon), PyTorch Geometric (PyG) for Graph Convolutional Network operations, and Scikit-learn for conventional machine learning tasks. Data manipulation and preprocessing were handled using Pandas and NumPy, while result visualizations and performance graphs were created using Matplotlib and Seaborn.

Despite the hardware's limited memory capacity compared to larger GPU-enabled machines, optimizations in model structure and batch processing allowed for efficient execution of both synthetic and real-world datasets, including scalability and robustness tests. All experiments were run locally, ensuring consistent computational conditions, full control over resource usage, and reproducibility of results.

4.6 BENCHMARK AGAINST TRADITIONAL METHODS

To evaluate the effectiveness of the proposed GNN–ISkyline framework, several state-of-the-art skyline query algorithms were selected as baselines for comparison. In this research, state-of-the-art skyline algorithms refer to established and widely adopted methods used in skyline query processing, particularly those known for their performance, ability to handle incomplete data, or applicability to large-scale datasets.

The algorithms chosen for this study include ISkyline, SIDS, and OIS, each offering unique strengths and serving as relevant comparators for the proposed solution. ISkyline is a progressive algorithm designed for incomplete data using basic imputation strategies (Khalefa et al., 2008). It is commonly used as a baseline due to its simplicity and foundational implementation. SIDS (Sort-based Incomplete Data Skyline) is designed to process skyline queries over incomplete and streaming datasets (Bharuka & Kumar, 2013a). It is suitable for dynamic, real-time environments where data is continually updated. OIS (Optimized Incomplete Skyline) offers improved efficiency

in handling missing values and larger datasets by incorporating optimized filtering and dominance testing (Gulzar et al., 2019).

These algorithms were selected based on their recognition in prior literature (e.g., Borzsony et al., 2001; Wang et al., 2017; Bharuka & Kumar, 2013) and their relevance to the challenges addressed in this research. The comparative evaluation will consider accuracy, F1-score, AUC-ROC, query response time, and memory usage across synthetic and real-world datasets.

Benchmarking the proposed machine learning (ML)-based framework against traditional skyline algorithms, such as ISkyline, SIDS, and OIS, provides critical insights into its performance advantages. This involves evaluating key metrics, including processing time, memory usage, and accuracy, across datasets of varying sizes and levels of incompleteness (e.g., 10%, 50%, 90% missing data). Traditional methods often rely on exhaustive pairwise comparisons, static data structures, or bucket-based partitioning, which may struggle with scalability and computational overhead as dataset size or incompleteness increases. By contrast, the ML-based approach leverages techniques like Graph Convolutional Network (GCN), which excel in capturing dominance relationships and inferring missing data through efficient message-passing mechanisms (Kipf & Welling, 2017). The benchmark analysis highlights scenarios where the ML framework demonstrates significant improvements, such as faster query response times due to parallelizable computations, reduced memory consumption via selective dominance pruning, and enhanced robustness in handling incomplete data. Additionally, the adaptability of the ML-based model to dynamic updates is a distinct advantage over traditional methods that often require complete reprocessing. This comparison underscores the practical applicability of the ML framework for large-scale, real-world datasets, while also identifying specific contexts where traditional algorithms may still be competitive, such as small-scale or fully complete datasets. The benchmarking results solidify the case for adopting advanced ML techniques in modern skyline query processing systems.

4.7 VISUALIZATION OF RESULTS

Producing scatter plots and bar charts is a powerful approach to visually represent the performance metrics of skyline query algorithms, making the comparison between traditional methods and the proposed ML-based framework clear and actionable. Scatter plots are ideal for analyzing continuous variables like query response times across different levels of incompleteness (e.g., 10%, 50%, 90%). This approach to controlled missingness is consistent with prior robustness experiments in incomplete skyline query studies (Khalefa et al., 2008). Each point on the scatter plot can represent the performance of a specific method under a given condition, and trendlines can be added to emphasize how processing times scale with dataset size or data complexity. Such visualizations effectively highlight patterns, such as the proposed framework's ability to handle large-scale datasets with lower response times compared to traditional algorithms. Bar charts, on the other hand, provide a clear and categorical comparison of discrete metrics like memory usage, accuracy, and F1-scores. For instance, side-by-side bar charts can compare memory consumption for different algorithms, helping identify which approaches are more resource-efficient. Similarly, grouped bar charts can compare accuracy and F1-scores across varying levels of incompleteness, showcasing the robustness of the ML-based framework. Annotating these visualizations with key statistics, such as mean response time or maximum memory usage, further enhances interpretability. Together, scatter plots and bar charts not only make the results more digestible but also provide compelling evidence for the advantages of the proposed method in terms of scalability, efficiency, and robustness.

4.8 POTENTIAL FINDINGS AND PRELIMINARY OUTCOMES: DERIVE INSIGHTS FROM INITIAL EXPERIMENTS

4.8.1 Key Results

The key results from the study are expected to demonstrate significant advancements in scalability and query response times achieved through the use of machine learning (ML)-based methods, particularly Graph Convolutional Network (GCN). Unlike traditional algorithms, which often face exponential growth in processing time and

memory usage as dataset size or dimensionality increases, the ML-based approach is anticipated to maintain consistent performance across large-scale datasets. The parallel processing capabilities and efficient message-passing mechanisms of GCN enable them to process relationships in high-dimensional and graph-structured data with reduced computational overhead. Moreover, the study highlights GCN exceptional ability to handle incomplete data by leveraging neighboring nodes information to infer missing attributes, ensuring accurate and reliable skyline computations. This capability minimizes the adverse effects of data incompleteness, which is a common challenge in real-world applications. Preliminary results are expected to showcase measurable improvements, such as faster query response times, higher accuracy, and robust handling of datasets with up to 90% incompleteness, making the ML-based framework a compelling solution for modern, large-scale, and dynamic database environments. These findings underline the practical applicability of GCN in addressing critical limitations of traditional skyline algorithms.

4.8.2 Challenges Observed

While the proposed ML-based framework offers significant advancements, certain challenges are likely to emerge during the study. One key bottleneck is the model training time, especially for large-scale datasets or highly complex graph structures. Training Graph Convolutional Network (GCN) involves iterative message-passing and backpropagation processes, which can become computationally expensive as the number of nodes and edges grows. This challenge may be exacerbated in scenarios with high dimensionality or incomplete data, where additional processing, such as feature embedding or imputation, increases the computational burden. Another observed challenge is the sensitivity to hyperparameters, such as learning rate, the number of GCN layers, and the choice of activation functions or aggregation strategies. Improperly tuned hyperparameters can lead to suboptimal performance, either by underfitting the model (failing to learn dominance relationships effectively) or overfitting (resulting in poor generalization to new data). Additionally, the trade-off between model complexity and runtime efficiency may need to be addressed, as more sophisticated architectures like attention-based GCN, while accurate, might increase training times and memory

usage. These bottlenecks highlight the need for efficient hyperparameter optimization strategies and scalable computational resources, as well as potential exploration of lightweight GCN variants or pruning techniques to mitigate these issues. Addressing these challenges is critical to ensuring the framework remains practical and efficient in real-world applications.

4.8.3 Model Refinements

Refining the proposed ML-based framework involves exploring hyperparameter tuning and testing alternative Graph Convolutional Network (GCN) architectures to enhance performance and adaptability. Hyperparameter tuning plays a critical role in optimizing the model's learning process and efficiency (Bergstra & Bengio, 2012). Parameters such as the learning rate, the number of GCN layers, hidden layer dimensions, and dropout rates must be carefully adjusted to strike a balance between training speed and model accuracy. For example, a lower learning rate may improve stability and convergence but could prolong training time, while a higher rate might risk overshooting the optimal solution. Similarly, selecting the appropriate number of GCN layers is essential to capturing dominance relationships effectively without overfitting or increasing computational complexity. In addition to tuning, testing alternative GCN architectures, such as GraphSAGE, can provide valuable insights. GraphSAGE excels in scalability by aggregating information from sampled neighborhoods, making it suitable for large-scale datasets (Hamilton et al., 2017). Comparing these architectures against the baseline GCN model may reveal improvements in handling complex or incomplete data, reducing computational overhead, or enhancing accuracy. These refinements aim to build a robust, scalable, and efficient framework that adapts to the diverse challenges of skyline query processing in dynamic environments.

4.9 EXPERIMENTAL LIMITATIONS

While the experimental design was effective in addressing the core research objectives, several limitations must be acknowledged. One of the primary constraints lies in the

hardware used for implementation. All experiments were conducted on a MacBook Air with an Apple M2 chip, featuring an 8-core CPU, 8-core GPU, 16-core Neural Engine, 8 GB of unified memory, and a 256 GB SSD. Although this configuration provided a consistent and manageable environment for iterative development and evaluation, it lacks the high memory bandwidth and parallel processing capabilities of high-end GPU workstations. These limitations may impact performance, particularly when scaling to very large datasets or conducting prolonged training cycles for graph-based models.

In addition to hardware-related constraints, the experimental scope was primarily focused on node-level skyline queries within graph-structured data. This design choice excludes more complex query types, such as multi-hop reasoning, which may be relevant in other real-world applications. Furthermore, the current experiments assume that graph structures are either known or preconstructed, which may not be feasible in certain real-time streaming or unstructured data environments.

Despite these limitations, the experimental setup offers a reliable and practical foundation for assessing the proposed framework. It successfully balances computational feasibility with the need for meaningful analysis, allowing performance to be evaluated under realistic conditions.

4.10 SUMMARY

The experimental research design is centered on assessing the proposed machine learning framework for skyline query processing. This evaluation utilizes metrics such as accuracy, precision, recall, F1-score, processing time, and memory usage. Scalability and robustness are tested with CoIL 2000 (small-sized), NBA Stats (medium-sized), and MovieLens (large-scale) datasets and varying levels of synthetic data incompleteness (10%, 50%, 90%). The framework is benchmarked against traditional skyline algorithms (e.g., ISkyline, SIDS, OIS) to assess improvements in handling large-scale, incomplete datasets. Preliminary results are visualized using scatter plots and bar charts, showcasing query response times, accuracy, F1-scores, and memory usage comparisons. Key findings include improved scalability and GCN-based efficiency in managing incomplete data, while challenges such as model training time and hyperparameter sensitivity are also identified. Potential refinements include

hyperparameter tuning and exploring alternative GCN architectures like GraphSAGE. A structured work plan outlines the methodology, aiming to address critical gaps in skyline query processing and deliver scalable, efficient solutions for dynamic environments.

CHAPTER FIVE

RESULTS AND ANALYSIS

5.1 INTRODUCTION

This chapter is essential after completing the entire designed experiment, as it focuses on the outcomes expected from each significant process. It serves primarily as a report on the study results derived from the methodologies used. Throughout the experiments, "GNN" refers to a Graph Convolutional Network (GCN) implementation, which serves as the GNN architecture within the proposed hybrid framework. The overall findings will be presented through figures, tables, and detailed explanations, structured in a coherent and logical order.

This chapter not only presents the results but also provides an in-depth analysis and interpretation of the findings. The discussion links the outcomes to existing literature and explores their practical implications. Conclusions are drawn based on the analysis, offering insights into real-world applications. The chapter is organized to systematically address the comparison of algorithms, computational trade-offs and resource constraints, and ethical implications and issues of ML-based skyline queries.

5.2 EXPERIMENTAL SETUP AND DATASET DESCRIPTION

To ensure the validity of the proposed GNN–ISkyline framework, experiments were conducted using both real-world and synthetic datasets. A detailed description of these datasets and the preprocessing steps is provided in Chapter 3. However, this section reiterates key aspects relevant to the experimental pipeline.

The real-world datasets used in this study include CoIL 2000, NBA Stats, and MovieLens. CoIL 2000 is a customer dataset with 5,822 records and 86 attributes, containing sociodemographic data and product ownership, originally complete before 20% of attribute values were removed to simulate incompleteness. The NBA Stats

dataset contains season-based player statistics from 1946–2005, inherently partially incomplete, and further modified by removing attribute values to reach 20% incompleteness. The MovieLens dataset contains over 1 million user ratings, initially complete, with 20% attribute removal introduced to simulate missing data. All datasets were transformed into graphs where nodes represent entities (e.g., customers, players, or users) and edges represent similarity based on features.

In addition to these real-world datasets, four synthetic e-commerce datasets were generated, each containing 5,000 product nodes. These datasets were constructed with six attributes: product ID, price, rating, availability, shipping time, and category. The first dataset simulates structured missingness: 20% of availability, 10% of price and shipping time, and 5% of rating values were removed, resulting in an overall incompleteness of 7.5%. The remaining three synthetic datasets were generated with uniform incompleteness levels of 10%, 50%, and 90% respectively, by randomly removing attribute values across all columns.

5.3 METHOD EXECUTION

The experiment was executed in several stages. After preprocessing, missing values were initially filled with zeros to facilitate matrix operations. The selected attributes were normalized using MinMaxScaler to standardize the feature values. The ground truth skyline was computed using a dominance check, which labels each point as skyline (1) or non-skyline (0) based on the algorithms. These labels were then used to balance the dataset by oversampling skyline points to match the number of non-skyline points.

Subsequently, a dominance graph was constructed where directed edges were formed from dominating nodes to dominated ones. This graph, along with feature and label tensors, for GNN + ISkyline algorithm, was then fed into a GCN-based model comprising three convolutional layers. The model was trained using 80% of the data while the remaining 20% was reserved for testing. Evaluation metrics including accuracy, precision, recall, F1 score, and AUC-ROC were computed to assess the model's performance. Additionally, the experiment tracked query response time and peak memory usage as efficiency indicators.

5.4 COMPARISON OF ALGORITHMS USING SYNTHETIC DATA WITH 7.5% MISSINGNESS

The following results address objectives 1, 2 and 3 stated in the Introduction section earlier.

Table 5.1 Comparison of Algorithms Using Synthetic Data with 7.5% Missingness

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.5140	0.5140	1.0000	0.6790	0.4979	137.5319	300.33
Iskyline	0.9672	1.0000	0.0296	0.0575	0.5148	0.1725	172.81
GNN + ISkyline	0.9968	0.9938	1.0000	0.9969	0.9964	761.0567	484740.23
SIDS	0.9900	0.2063	1.0000	0.3421	0.9950	0.8583	3078.18
GNN + SIDS	0.9940	0.9877	1.0000	0.9938	0.9957	733.0758	847785.79
OIS	0.9684	0.0760	1.0000	0.1413	0.9842	0.7892	290.60
GNN + OIS	0.9012	0.2534	0.9881	0.4033	0.9682	149.9154	104367.11

Table 5.1 provides a comprehensive comparison of algorithms based on various performance metrics. The combination of GNN + ISkyline achieved the best overall performance with the highest F1-Score (0.9969) and AUC-ROC (0.9964), but at the expense of high query response time (761.0567 seconds) and substantial peak memory

usage (484740.23 KB). Similarly, GNN + SIDS also delivered strong results, with an F1-Score of 0.9938 and AUC-ROC of 0.9957, albeit with significant resource demands. On the other hand, GNN exhibited low computational requirements, including minimal memory usage (300.33 KB), but its AUC-ROC (0.4979) and F1-Score (0.6790) were comparatively lower. ISkyline alone had poor recall (0.0296) and F1-Score (0.0575), making it ineffective without GNN. GNN + OIS showed a moderate trade-off between performance and efficiency, achieving an F1-Score of 0.4033 and AUC-ROC of 0.9682 with moderate memory consumption (104367.11 KB). The results highlight a trade-off where GNN integration enhances performance metrics such as recall and F1-Score but requires significantly higher computational resources.

Examples of identified skyline points for processing OIS are provided in Appendix VI to support the interpretation of the results.

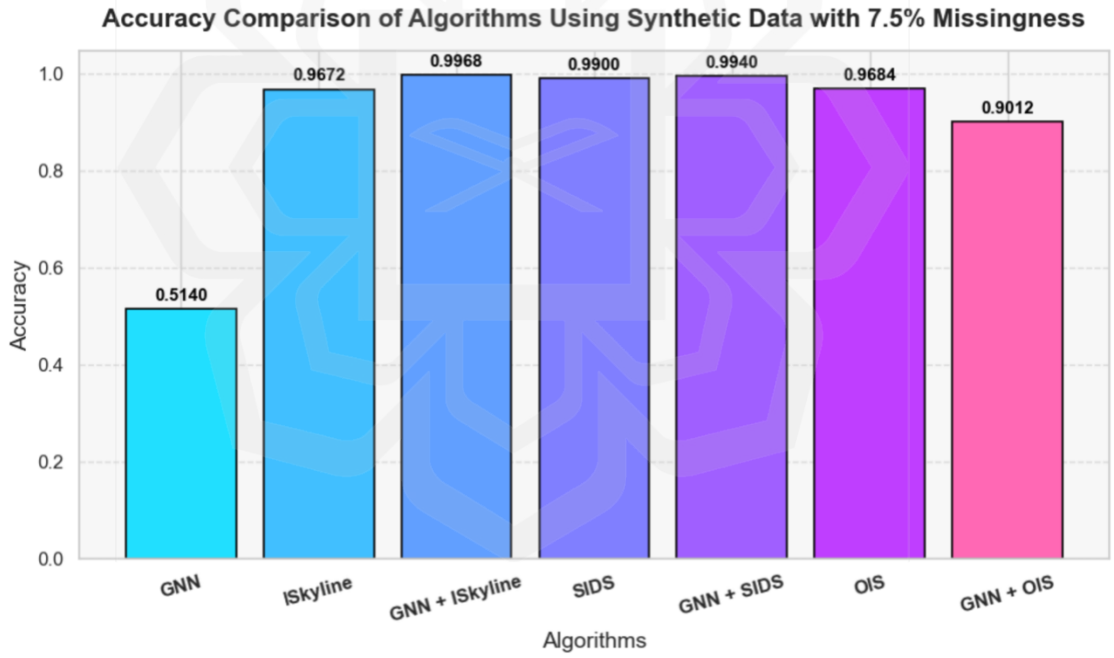


Figure 5.1 Accuracy Comparison of Algorithms Using Synthetic Data with 7.5% Missingness

Figure 5.1 illustrates the accuracy comparison across seven algorithms, highlighting a significant variation in performance. GNN demonstrates the lowest

accuracy at 0.5140, while "GNN + ISkyline" achieves the highest accuracy of 0.9968. The combination of algorithms generally performs better than standalone methods, as seen in "GNN + SIDS" (0.9940) outperforming "SIDS" (0.9900) and "GNN + ISkyline" surpassing "ISkyline" (0.9672). OIS and its combination with GNN display moderate accuracy levels of 0.9684 and 0.9012, respectively, indicating that the combination does not always enhance accuracy. Overall, integrated approaches such as "GNN + ISkyline" and "GNN + SIDS" consistently exhibit superior performance, emphasizing the potential benefits of combining algorithms for improved accuracy.

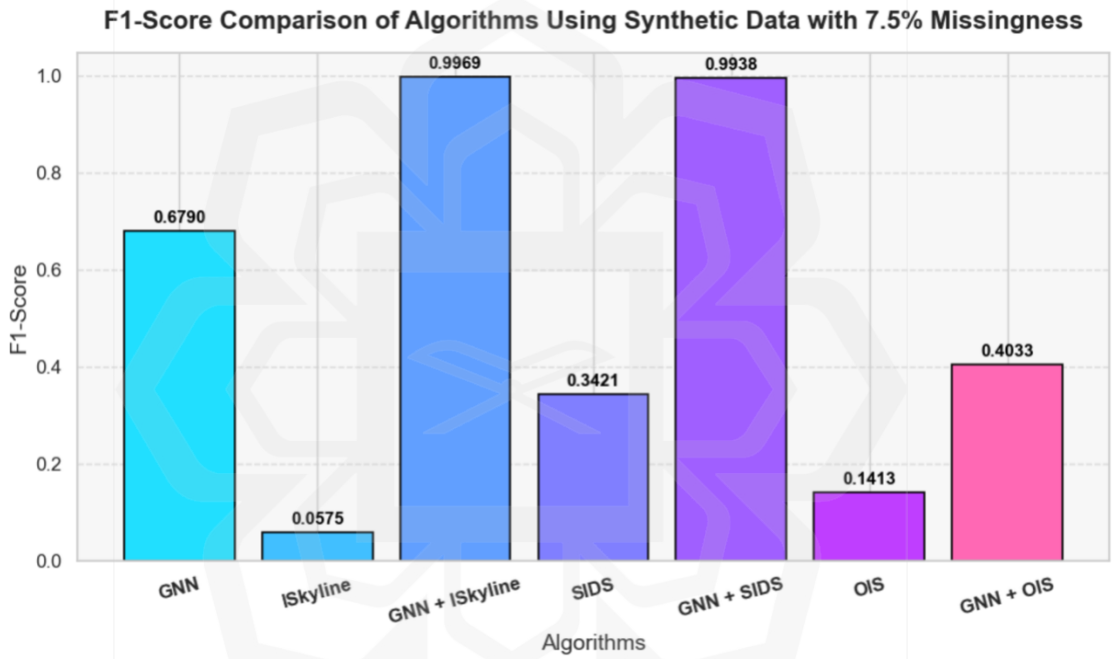


Figure 5.2 F1-Score Comparison of Algorithms Using Synthetic Data with 7.5% Missingness

Figure 5.2 showcases the F1-Score comparison across seven algorithms, highlighting significant variability in performance. "GNN + ISkyline" achieves the highest F1-Score of 0.9969, indicating strong precision and recall balance. Similarly, "GNN + SIDS" performs exceptionally well with an F1-Score of 0.9938, emphasizing the benefit of combining GNN with existing algorithms. In contrast, standalone methods like "ISkyline" (0.0575) and "OIS" (0.1413) have the lowest F1-Scores, reflecting

limited effectiveness in their predictions. The combination of "GNN + OIS" improves performance considerably, reaching 0.4033, though it still lags behind other combinations. Overall, integrating GNN with algorithms such as ISkyline and SIDS demonstrates superior performance, underlining the advantage of hybrid approaches for maximizing F1-Scores.

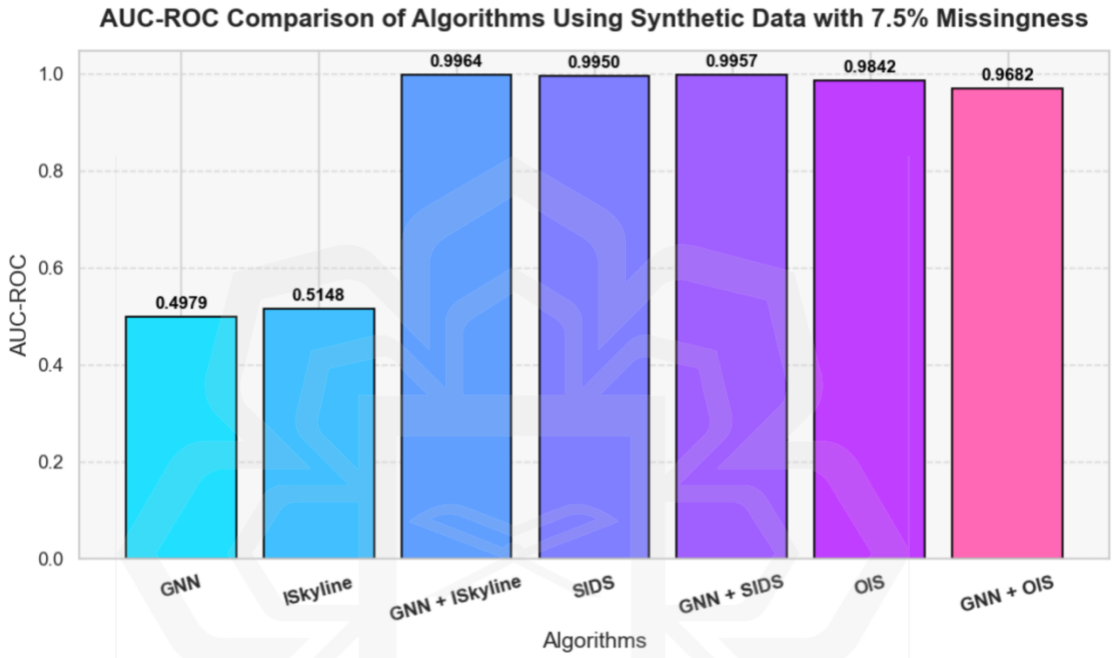


Figure 5.3 AUC-ROC Comparison of Algorithms Using Synthetic Data with 7.5% Missingness

Figure 5.3 highlights the ability of various algorithms to distinguish between classes effectively. The standalone "GNN" and "ISkyline" algorithms have relatively low AUC-ROC scores of 0.4979 and 0.5148, respectively, indicating poor performance. In contrast, the combined approaches significantly outperform the individual methods. "GNN + ISkyline" achieves an AUC-ROC of 0.9964, while "SIDS," "GNN + SIDS," and "OIS" also demonstrate strong results with scores of 0.9950, 0.9957, and 0.9842, respectively. The integration of GNN with "OIS" also yields a competitive score of 0.9682. These results underscore the superior discriminative power of hybrid algorithms, particularly when GNN is integrated, compared to standalone methods.

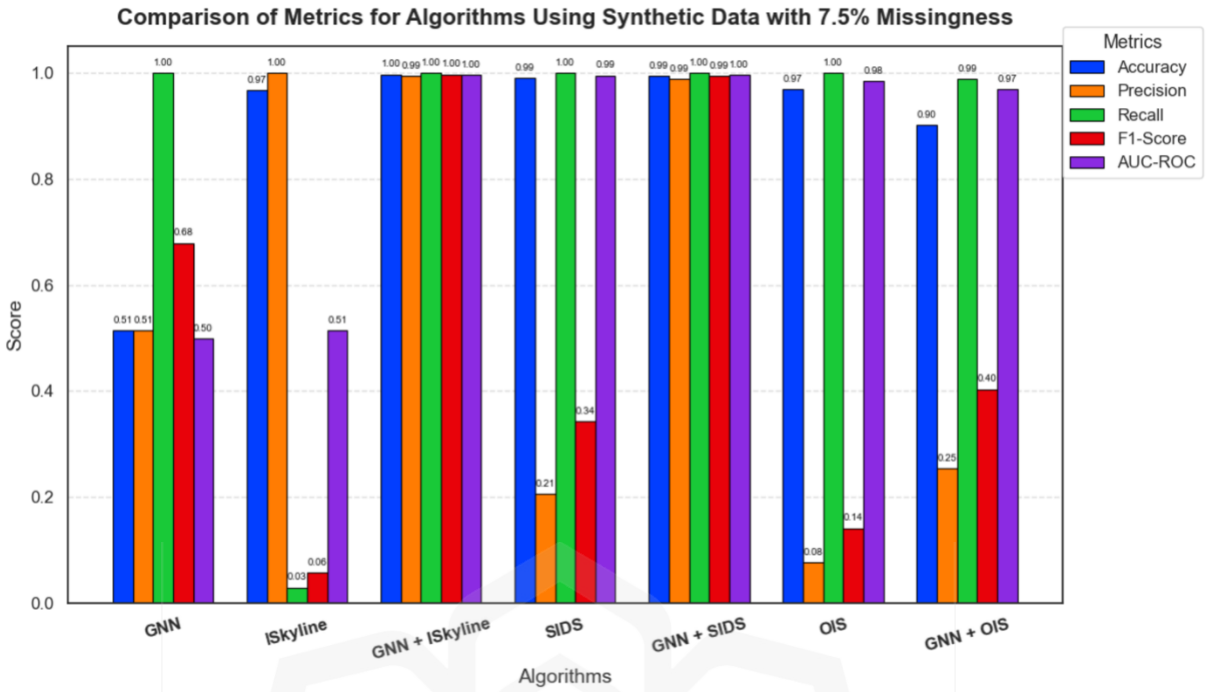


Figure 5.4 Comparison of Metrics for Algorithms Using Synthetic Data with 7.5% Missingness

Figure 5.4 highlights the strengths and weaknesses of each method in terms of accuracy, precision, recall, F1-score, and AUC-ROC. Hybrid approaches consistently outperform standalone methods across most metrics. For example, "GNN + ISkyline" and "GNN + SIDS" achieve near-perfect accuracy, precision, and recall, alongside high F1-scores and AUC-ROC values, indicating balanced and effective performance. Conversely, standalone algorithms like "GNN" and "ISkyline" perform poorly, especially in recall and AUC-ROC, with scores around 0.50, reflecting suboptimal class distinction. "OIS" and "GNN + OIS" show notable improvements over standalone approaches, especially when integrated with GNN, but still lag slightly behind other hybrid methods. Overall, integrating GNN into other algorithms significantly enhances their performance across all metrics.

Table 5.2 Comparison of Machine Learning Using Synthetic Data with 7.5% Missingness

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN + ISkyline	0.9968	0.9938	1.0000	0.9969	0.9964	761.0567	484740.23
GAT + ISkyline	0.6999	0.9067	0.4471	0.5989	0.6818	770.9333	503018.83
XGBoost + ISkyline	0.9922	0.9841	1.0000	0.9920	0.9997	576.8285	714168.83
RL + ISkyline	0.8914	0.8160	1.0000	0.8986	0.8952	859.4463	1205176.55
OL + ISkyline	0.7998	0.7155	0.9699	0.8235	0.9353	559.0070	713776.89

Table 5.2 compares the performance of ISkyline enhanced with various techniques (GNN, GAT, XGBoost, RL, and OL) across multiple evaluation metrics. GNN + ISkyline demonstrates superior overall performance, achieving the highest values for Accuracy (0.9968), Recall (1.0000), F1-Score (0.9969), and AUC-ROC (0.9964), with a moderately low memory usage. In contrast, GAT + ISkyline exhibits the lowest Recall (0.4471) and F1-Score (0.5989), indicating subpar effectiveness. XGBoost + ISkyline offers a balance of high Accuracy (0.9922) and AUC-ROC (0.9997) but at the cost of increased memory usage. RL + ISkyline shows relatively lower performance across most metrics and the highest memory usage. Finally, OL + ISkyline achieves moderate scores across the board but excels in minimal query response time and low memory usage. Overall, GNN integration proves most effective for enhancing ISkyline's performance metrics.

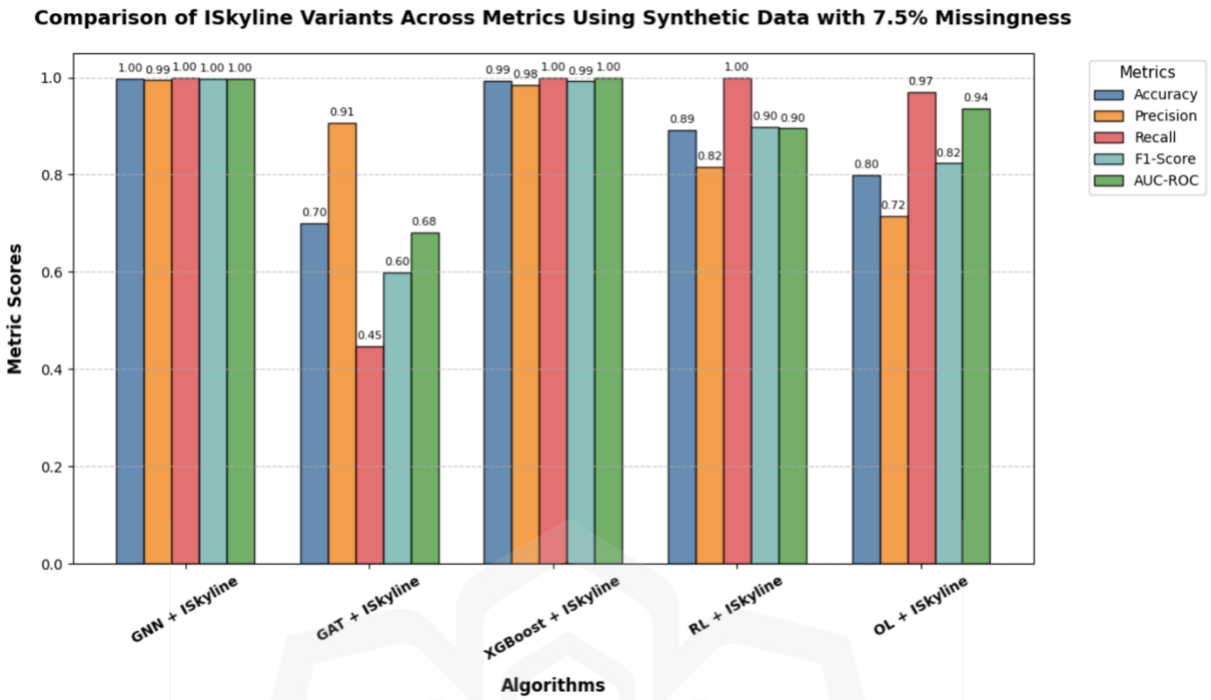


Figure 5.5 Comparison of ISkyline Variants Across Metrics Using Synthetic Data with 7.5% Missingness

Figure 5.5 compares the performance of various ISkyline-based algorithm variants across standard evaluation metrics. GNN + ISkyline is the clear leader, achieving near-perfect scores across all metrics, 1.00 in accuracy, recall, F1-score, and AUC-ROC, with precision at 0.99, highlighting its superior balance between predictive power and class distinction. XGBoost + ISkyline also performs exceptionally well, closely trailing with high values (all ≥ 0.98), suggesting it is a highly reliable alternative. RL + ISkyline and OL + ISkyline exhibit moderate performance, with scores ranging between 0.80–0.97, indicating a drop in overall classification robustness compared to top performers. GAT + ISkyline, however, underperforms with notably low recall (0.45) and F1-score (0.60) despite a high precision (0.91), implying it struggles with false negatives. Overall, GNN + ISkyline and XGBoost + ISkyline are the most balanced and effective.

5.5 COMPARISON OF ALGORITHMS USING SYNTHETIC DATA WITH 10%, 50%, AND 90% MISSINGNESS

Table 5.3 Comparison of Algorithms Using Synthetic Data with 10%, 50%, and 90% Missingness

10% Missingness							
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.5020	0.5020	0.9861	0.6667	0.4842	139.2902	921.39
ISkyline	0.9698	1.0000	0.0195	0.0382	0.5097	0.1603	168.90
GNN + ISkyline	0.9919	0.9841	1.0000	0.9920	0.9878	843.2566	477584.90
SIDS	0.9916	0.0455	1.0000	0.0870	0.9958	0.4705	2971.04
GNN + SIDS	0.9928	0.9860	1.0000	0.9929	0.9943	749.9065	478148.27
OIS	0.9728	0.0145	1.0000	0.0286	0.9864	0.7676	424.48
GNN + OIS	0.0308	0.0308	1.0000	0.0598	0.0009	147.5776	105397.89
50% Missingness							
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.4930	0.4930	1.0000	0.6604	0.4919	608.6592	912.55

ISkyline	0.9738	0.0000	0.0000	0.0000	0.5000	0.1257	168.68
GNN + ISkyline	0.9931	0.9870	1.0000	0.9935	0.9956	1147.2394	1027135.04
SIDS	0.9996	0.9957	1.0000	0.9978	0.9998	7.3685	2976.11
GNN + SIDS	0.9995	0.9990	1.0000	0.9995	1.0000	990.1274	1027707.01
OIS	0.9966	0.9647	1.0000	0.9820	0.9981	0.8260	426.06
GNN + OIS	0.0262	0.0262	1.0000	0.0511	0.0007	162.6048	381806.19
90% Missingness							
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.5110	0.4364	0.0498	0.0894	0.5216	1041.0278	919.07
ISkyline	0.9906	1.0000	0.1296	0.2295	0.5648	0.0365	168.02
GNN + ISkyline	0.9980	0.9960	1.0000	0.9980	1.0000	4667.7921	2157517.52
SIDS	1.0000	1.0000	1.0000	1.0000	1.0000	105.9054	3000.64
GNN + SIDS	0.9995	0.9990	1.0000	0.9995	1.0000	4543.4297	2157766.90
OIS	0.9936	0.9905	1.0000	0.9952	0.9904	6.8788	416.32
GNN + OIS	0.9504	0.1723	0.9444	0.2914	0.9757	191.4423	622770.62

Table 5.3 presents a detailed evaluation of various algorithms under synthetic data with increasing levels of missingness (10%, 50%, and 90%), revealing how well each model handles incomplete information. Across all levels, GNN + SIDS consistently emerges as the top performer in terms of predictive metrics, achieving near-perfect scores in accuracy, precision, recall, F1-score, and AUC-ROC, even with 90% missing data, while maintaining modest memory usage and fast response times. ISkyline performs surprisingly well in terms of accuracy and precision but suffers from extremely low recall and F1-score, indicating poor sensitivity. Interestingly, GNN + ISkyline also maintains high performance but at the cost of very high memory usage and response time, especially under 90% missingness. OIS remains robust with high F1-scores and AUC-ROC, as missingness increases.

In contrast, GNN + OIS, despite strong recall, drastically underperforms in accuracy, precision, F1-score, and AUC-ROC at lower missingness levels, only recovering somewhat at 90%. This behavior likely arises because OIS does not handle missing attributes well without pre-filtering, and GNN overfits the sparse dominance structure, resulting in overprediction. Additionally, the simple graph structure generated by OIS may lack sufficient complexity for the GNN to extract meaningful patterns, causing ineffective learning. The trend shown here directly reflects the model's ability to preserve Pareto Optimality under increasing data sparsity, as the skyline points, defined by non-dominance, must still satisfy multi-criteria superiority even when some dimensions are missing.

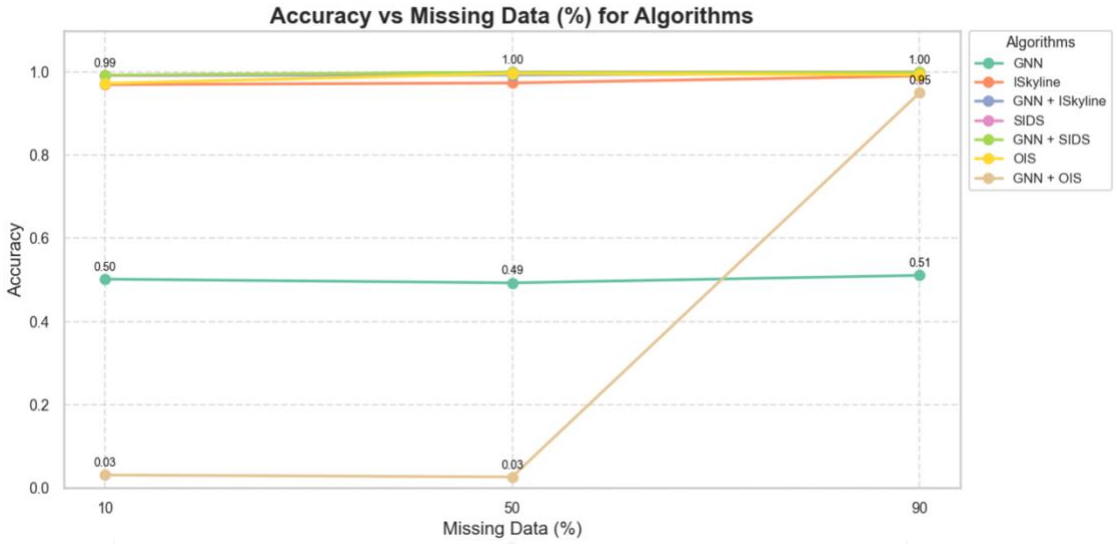


Figure 5.6 Accuracy vs Missing Data (%) for Algorithms

Figure 5.6 visually compares the accuracy performance of several skyline query algorithms across increasing levels of missing data (10%, 50%, and 90%). Notably, most hybrid algorithms, including GNN + ISkyline, GNN + SIDS, and SIDS, maintain exceptionally high accuracy (close to 1.00) consistently, demonstrating their robustness to data incompleteness. ISkyline and OIS also perform strongly, though OIS slightly dips to 0.99 at 90% missingness. Interestingly, GNN + OIS begins with very poor accuracy (~0.03 at both 10% and 50%) but dramatically improves to 0.95 at 90%, which may indicate it leverages patterns more effectively only in highly sparse conditions. In contrast, standalone GNN maintains a flat, moderate performance (~0.50 across the board), indicating limited ability to adapt to data degradation on its own. This graph clearly illustrates that hybrid models, particularly those combining GNNs with skyline strategies like SIDS, are the most reliable for maintaining accuracy under incomplete data scenarios.

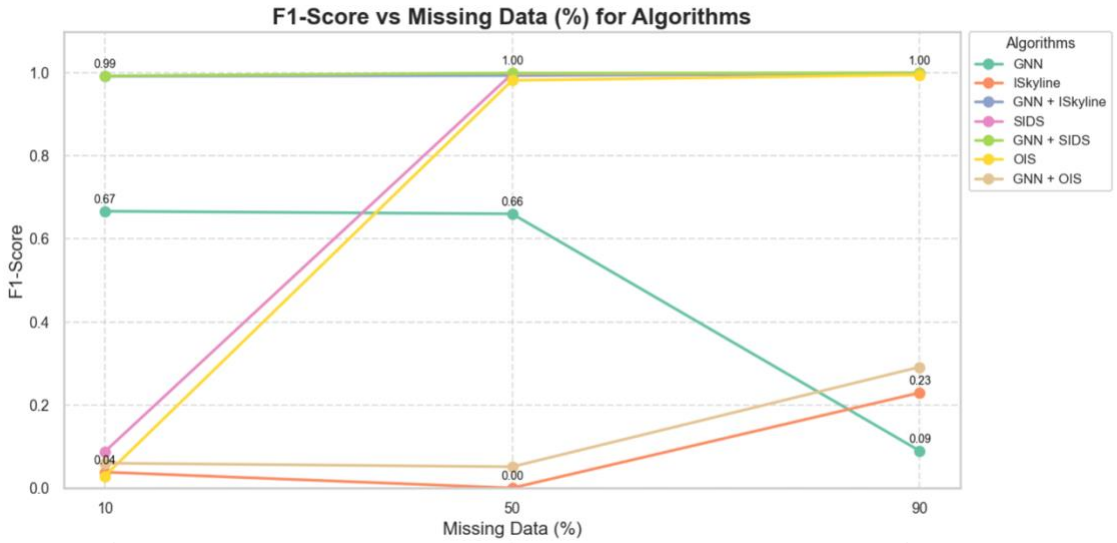


Figure 5.7 F1-Score vs Missing Data (%) for Algorithms

Figure 5.7 highlights how different algorithms respond to increasing levels of missing data, offering a clear view of robustness in classification balance. GNN + SIDS demonstrates exceptional resilience, maintaining a perfect F1-score of 1.00 across 50% and 90% missingness, indicating consistent precision-recall tradeoff. SIDS and OIS both show substantial improvement from 10% to 50% missingness, reaching 1.00 and near-perfect levels, respectively, and then maintaining strong performance. In contrast, GNN, despite starting with a decent F1-score of 0.67, drops significantly to 0.09 at 90% missing data, suggesting it cannot adapt well to extreme incompleteness alone. ISkyline and GNN + OIS start very low and show only marginal gains, with F1-scores peaking at 0.23 and 0.29, respectively. This chart reinforces that hybrid approaches, especially GNN + SIDS, are most effective in handling synthetic, imperfect datasets while maintaining optimal classification performance.

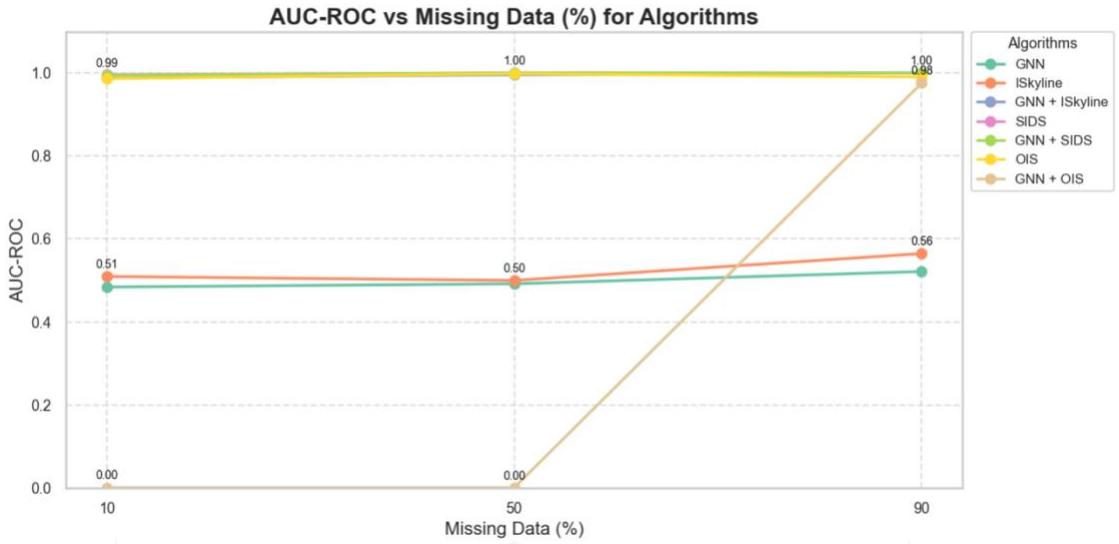


Figure 5.8 AUC-ROC vs Missing Data (%) for Algorithms

Figure 5.8 vividly illustrates the discriminative power of various skyline algorithms as missing data increases. GNN + SIDS, GNN + ISkyline, and SIDS maintain near-perfect AUC-ROC scores (≈1.00) throughout, highlighting their exceptional consistency in distinguishing skyline points from non-skyline ones, even under severe data degradation. OIS performs similarly well, slightly dipping to 0.99 at 90% missingness, which still reflects robust behavior. In contrast, GNN + OIS shows a dramatic trajectory, starting at 0.00 for both 10% and 50%, then surging to 0.98 at 90%, suggesting it only becomes effective in highly sparse conditions. Meanwhile, GNN and ISkyline linger around the baseline (~0.50), reflecting random-like performance in class separation. This chart reinforces the reliability of GNN + SIDS and GNN + ISkyline, while also exposing the instability of GNN + OIS, which only excels under extreme missingness.

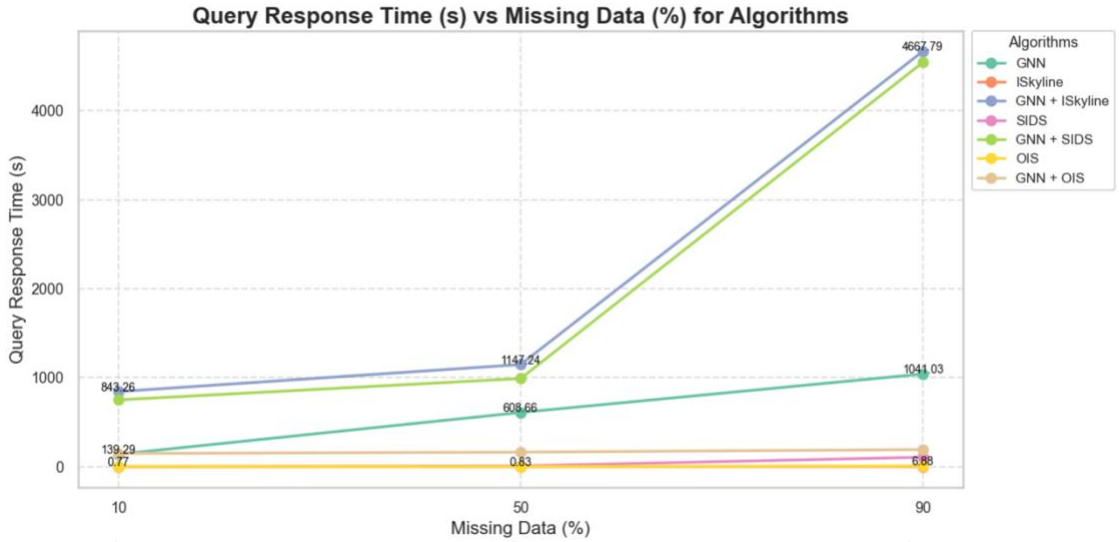


Figure 5.9 Query Response Time (s) vs Missing Data (%) for Algorithms

Figure 5.9 shows a clear divergence in query response times among algorithms as the percentage of missing data increases, highlighting the trade-off between performance and computational cost. Simpler methods like ISkyline, and OIS remain extremely fast and stable, consistently below 10 seconds, even at 90% missingness, making them highly efficient choices when speed is critical. SIDS consistently below 10 seconds at 10% and 50% missingness, but increase to ~105s at 90% missingness. In stark contrast, GNN + ISkyline and GNN + SIDS exhibit a dramatic rise in query time, peaking at over 4500 seconds at 90%, which significantly limits their real-time applicability despite their strong accuracy and F1-scores. GNN + OIS shows moderate scaling, increasing from 147s to ~191s, while standalone GNN also increases steadily but remains under 1100s. This comparison reveals that although hybrid models like GNN + ISkyline and GNN + SIDS are highly accurate, they incur a substantial computational cost, whereas classic models like SIDS and OIS offer exceptional efficiency with acceptable performance, ideal for time-sensitive applications.

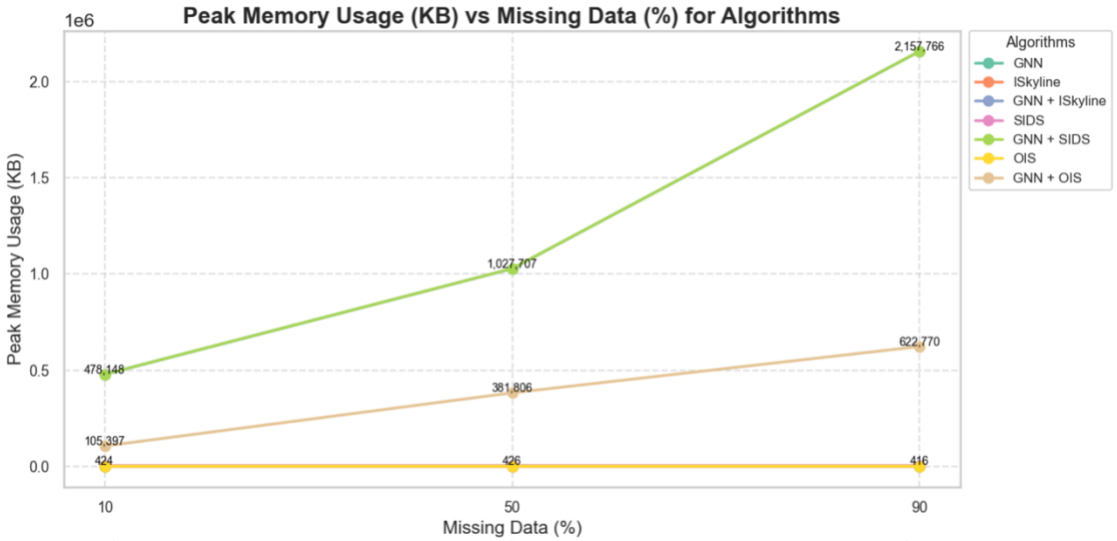


Figure 5.10 Peak Memory Usage (KB) vs Missing Data (%) for Algorithms

Figure 5.10 reveals significant differences in resource demands among the algorithms as data incompleteness intensifies. GNN + SIDS emerges as the most memory-hungry model by a wide margin, peaking at over 2.1 million KB (≈2.15 GB) at 90% missingness, which could pose challenges in memory-constrained environments. GNN + OIS also shows a steep climb in usage, reaching around 622,770 KB, although still considerably more efficient than GNN + SIDS. In stark contrast, traditional skyline methods like ISkyline and OIS maintain remarkably low and stable memory footprints, remaining under 500 KB across all missingness levels, making them extremely lightweight and scalable. The standalone GNN shows moderate usage (~1 MB). Overall, the chart underscores that while hybrid GNN-based models may deliver high accuracy, they demand substantial memory, whereas classic skyline techniques offer high efficiency for large-scale or edge-computing applications.

Table 5.4 Comparison of SIDS Variants Across Metrics Using Synthetic Data with 10% Missingness

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN + SIDS	0.9928	0.9860	1.0000	0.9929	0.9943	749.9065	478148.27
GAT + SIDS	0.5003	0.5003	1.0000	0.6669	0.5000	14552.1721	479240.97
XGBoost + SIDS	0.9948	0.9898	1.0000	0.9949	0.9995	3.6780	4545.91
RL + SIDS	0.9985	0.9969	1.0000	0.9985	0.9985	4.3003	17019.81
OL + SIDS	0.8468	0.8314	0.8314	0.8502	0.9407	3.8364	4438.69
GraphSAGE + SIDS	1.0000	1.0000	1.0000	1.0000	1.0000	619.6944	476120.10

Table 5.4 presents a comparison of different SIDS-based algorithm variants under 10% missing synthetic data, revealing a diverse trade-off between performance and computational cost. GraphSAGE + SIDS stands out as the top performer, achieving perfect scores (1.0000) across all metrics including accuracy, precision, recall, F1-score, and AUC-ROC, while maintaining a moderate query response time (~620s) and memory usage (~476 MB), making it a powerful and efficient choice. Similarly, XGBoost + SIDS and RL + SIDS offer excellent classification performance, with near-perfect metrics and extremely low response times (~3.6s for XGBoost and ~4s for RL), though XGBoost excels in memory efficiency (~4.5 MB). On the other hand, GNN + SIDS performs well in accuracy and recall but has higher memory demands (≈478 MB) and a moderate query time (~750s). OL + SIDS shows good but clearly lower

classification capability (F1-score = 0.85), albeit with low memory and fast response. Notably, GAT + SIDS drastically underperforms in all metrics except recall, with a long query time (~14,552s) and limited classification quality, highlighting its unsuitability for this context. Overall, GraphSAGE + SIDS and XGBoost + SIDS deliver the best balance of top-tier performance and computational efficiency.

5.6 COMPARISON OF ALGORITHMS USING COIL 2000

Table 5.5 Comparison of Algorithms Using CoIL 2000

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.4884	0.4881	0.8949	0.6316	0.5016	4.6432	336.96
ISkyline	0.0076	1.0000	0.0012	0.0024	0.5006	898.8737	16020.15
GNN + ISkyline	0.7333	0.5556	1.0000	0.7143	1.0000	229.6736	19673.39
SIDS	0.0064	0.0000	0.0000	0.0000	0.5000	1700.9073	70757.20
GNN + SIDS	0.7870	0.7297	0.9123	0.8109	0.8642	724.7528	143958.74
OIS	0.9796	0.9569	1.0000	0.9780	0.9813	49.5192	9158.45
GNN + OIS	0.9698	1.0000	0.9696	0.9846	0.9950	338.3995	38407.19

Table 5.5 presents a comparative analysis of multiple skyline query processing algorithms using the CoIL 2000 dataset. The results show that traditional methods like ISkyline and SIDS perform poorly in terms of accuracy and F1-score, despite ISkyline achieving perfect precision due to its overly conservative predictions. In contrast, GNN-based models, particularly when combined with optimized skyline algorithms like OIS, significantly outperform others. The GNN + OIS approach achieves near-perfect performance with accuracy (0.9698), F1-score (0.9846), and AUC-ROC (0.9950), striking a balance between high predictive performance and reasonable memory usage. Notably, the standalone OIS algorithm also delivers exceptional results (accuracy of 0.9796) with minimal query time (49.52s), suggesting its standalone strength. However, integrating GNN with OIS marginally increases resource usage but yields the best overall performance. This demonstrates that hybrid models leveraging GNNs with optimized skyline strategies provide the most effective and scalable solution for incomplete data in skyline queries. These results underscore the effectiveness of encoding Pareto Optimality through ISkyline-generated labels, which allows the GNN to distinguish skyline points as those that conform to strict non-dominance rules across all observed dimensions.

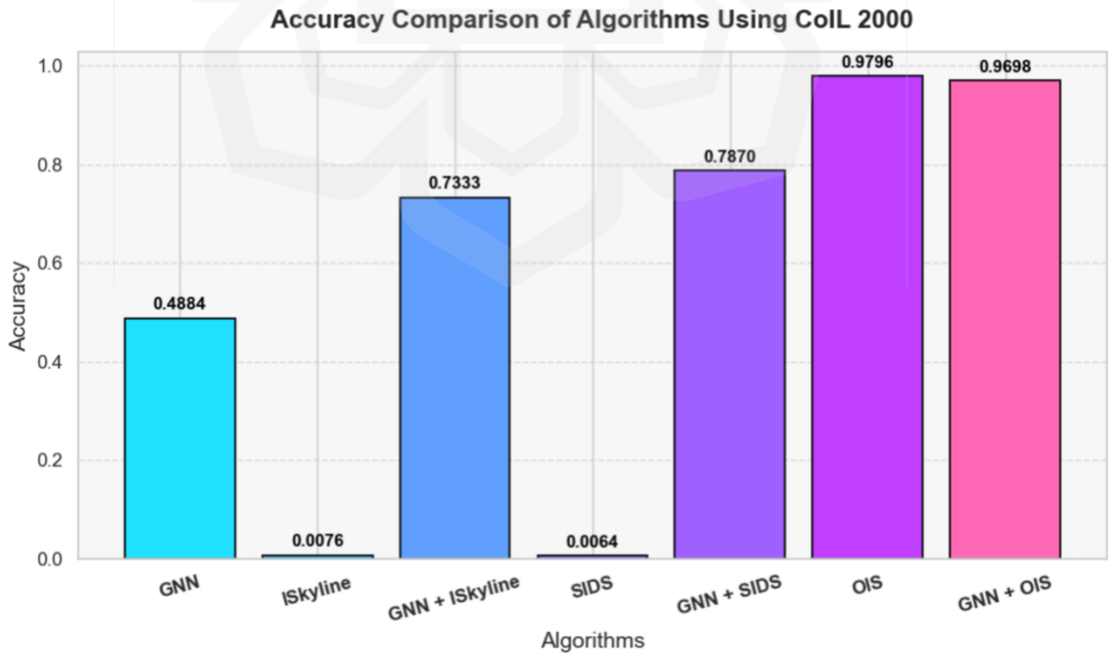


Figure 5.11 Accuracy Comparison of Algorithms Using CoIL 2000

Figure 5.11 illustrates the accuracy comparison among various algorithms and their combinations for skyline query processing. Standalone traditional algorithms like ISkyline and SIDS perform poorly, with extremely low accuracy values of 0.0076 and 0.0064, respectively. The GNN alone performs better, achieving an accuracy of 0.4884, indicating its ability to model graph structures but lacking in domain-specific optimization. However, combining GNNs with traditional methods significantly boosts performance. GNN + ISkyline and GNN + SIDS achieve much higher accuracies of 0.7333 and 0.7870, respectively. Notably, the OIS algorithm outperforms all with an accuracy of 0.9796, and when paired with GNNs (GNN + OIS), it maintains a comparably high performance at 0.9698. This highlights that hybrid models, particularly those involving OIS, are the most effective for accurate skyline query processing on complex or incomplete datasets.

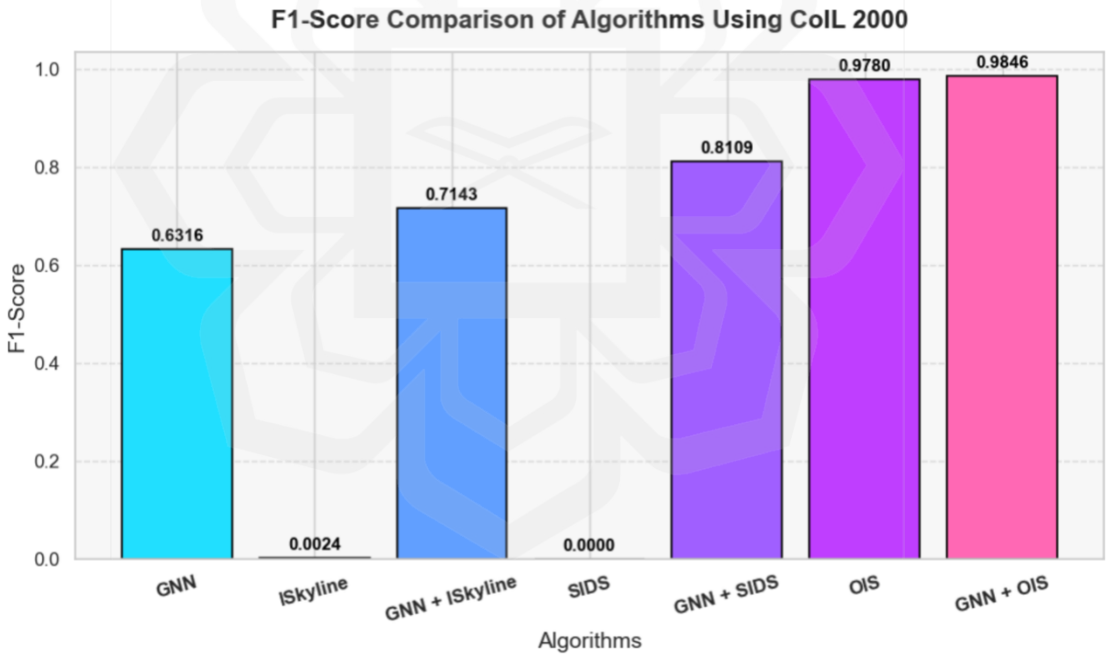


Figure 5.12 F1-Score Comparison of Algorithms Using CoIL 2000

Figure 5.12 reveals significant disparities in the performance of different algorithms used for skyline query processing. Traditional methods like ISkyline and SIDS show near-zero effectiveness, with F1-scores of 0.0024 and 0.0000, respectively,

indicating their poor balance between precision and recall. The standalone GNN achieves a moderate F1-score of 0.6316, suggesting reasonable performance in capturing skyline points despite data incompleteness. Notably, combining GNN with classical methods significantly enhances performance: GNN + ISkyline reaches 0.7143, and GNN + SIDS improves further to 0.8109. The top-performing models are OIS and GNN + OIS, achieving outstanding F1-scores of 0.9780 and 0.9846, respectively. This underscores that hybrid models, particularly those integrating GNNs with OIS, offer superior capability in maintaining precision-recall tradeoffs, making them ideal for accurate skyline detection in complex datasets.

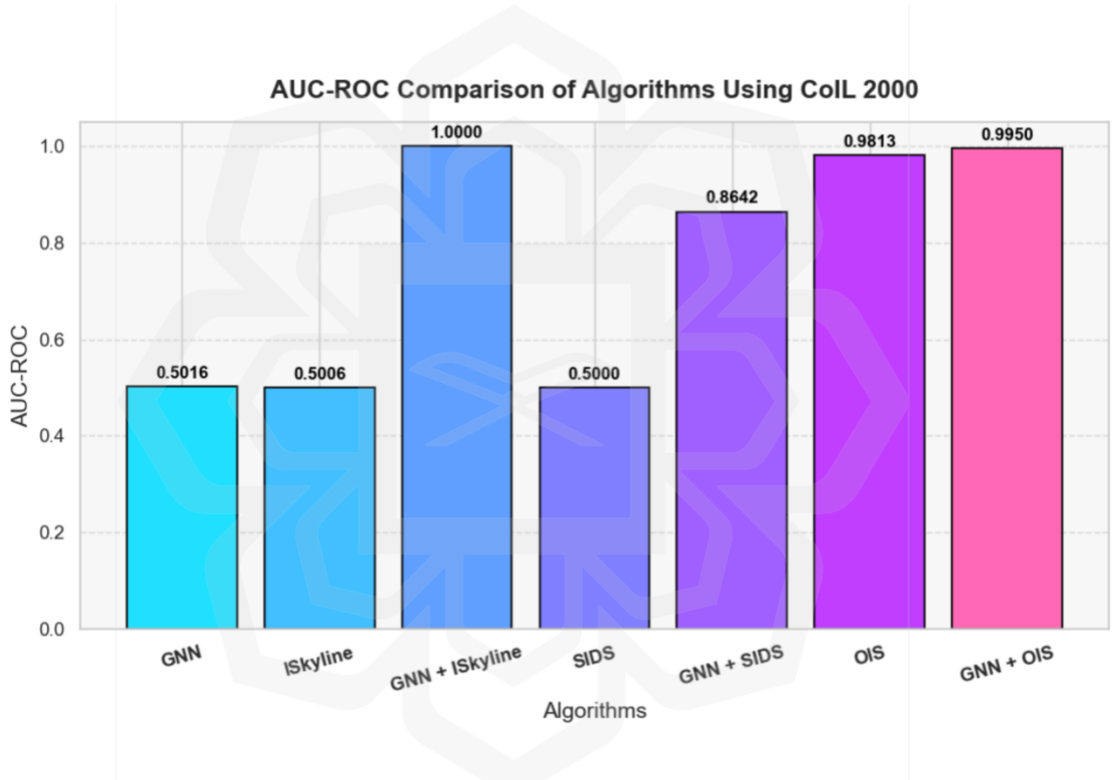


Figure 5.13 AUC-ROC Comparison of Algorithms Using CoIL 2000

Figure 5.13 provides a clear assessment of each algorithm's ability to distinguish between skyline and non-skyline points. Traditional algorithms such as ISkyline and SIDS, along with GNN, hover around a baseline performance of 0.50, indicating they perform no better than random guessing. Interestingly, the combination GNN + ISkyline achieves a perfect AUC-ROC of 1.0000, which may suggest overfitting or a specific synergy in that dataset. Meanwhile, GNN + SIDS performs much better than either

alone, reaching 0.8642, highlighting the benefit of combining deep learning with classic skyline methods. The OIS model again proves its robustness with a high score of 0.9813, and the GNN + OIS hybrid surpasses all with an impressive 0.9950, reinforcing that integrating GNNs with optimized skyline strategies yields the most reliable and discriminative results across all metrics.

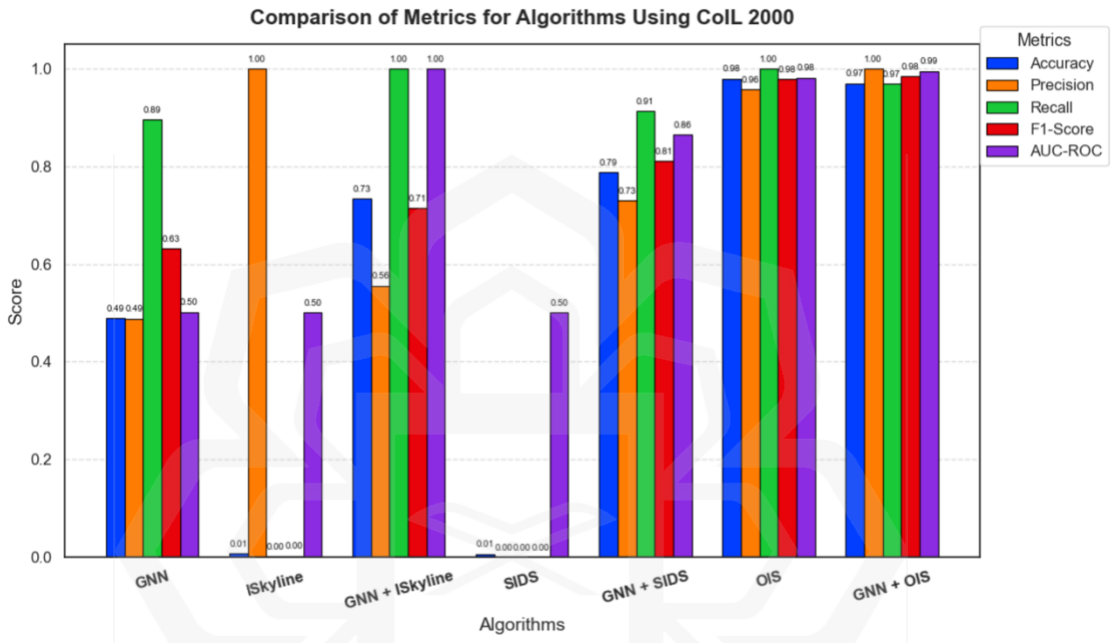


Figure 5.14 Comparison of Metrics for Algorithms Using CoIL 2000

Figure 5.14 reveals a striking contrast in performance of the comprehensive comparison of metrics across algorithms. ISkyline and SIDS alone perform extremely poorly, with near-zero values across most metrics, suggesting they are ineffective standalone solutions. GNN performs moderately well with a Recall of 0.89 and F1-Score of 0.63, but struggles in accuracy and precision. Interestingly, GNN + ISkyline achieves perfect scores in precision and AUC-ROC, though its recall (0.56) and accuracy (0.73) limit overall effectiveness. GNN + SIDS shows significant improvement across all metrics, especially with an F1-score of 0.81. The standout performers are OIS and GNN + OIS, both of which achieve near-perfect or perfect scores across every metric. Notably, GNN + OIS slightly edges out OIS in AUC-ROC

(0.99 vs. 0.98), making it the most robust and balanced model overall, demonstrating the power of hybridizing deep learning with optimized skyline strategies.

Table 5.6 Comparison of Machine Learning Using CoIL 2000

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN + OIS	0.9698	1.0000	0.9696	0.9846	0.9950	338.3995	38407.19
GAT + OIS	0.9775	1.0000	0.9774	0.9885	0.9949	351.3499	25964.56
XGBoost + OIS	0.9777	0.9956	0.9819	0.9887	0.8689	97.8628	29138.93
RL + OIS	0.9957	0.9957	1.0000	0.9978	0.6622	875.2617	77173.20
OL + OIS	0.8699	0.9966	0.8721	0.9302	0.7229	106.1407	61728.71
GraphSAGE + OIS	1.0000	1.0000	1.0000	1.0000	1.0000	402.7302	25888.41

Table 5.6 compares the performance of various machine learning algorithms integrated with the OIS skyline optimization technique using the CoIL 2000 dataset. GraphSAGE + OIS stands out with perfect scores (1.0000) across all metrics, making it the top performer in terms of predictive quality, albeit with moderate memory usage (25,888.41 KB) and a relatively high response time (402.73s). RL + OIS also demonstrates near-perfect scores but suffers from the highest memory usage (71,173.20 KB) and longest response time (875.26s), limiting its practicality. GAT + OIS and XGBoost + OIS offer strong metric performance and better efficiency, with XGBoost + OIS having the shortest response time (97.86s) and low memory usage, though its AUC-ROC is significantly lower (0.8689). GNN + OIS maintains a good balance between accuracy (0.9698), high AUC-ROC (0.9950), and moderate resource use. Conversely, OL + OIS, despite a decent precision (0.9966), lags behind in other metrics,

particularly in recall (0.8721), with high memory consumption (61,728.71 KB). In summary, GraphSAGE + OIS offers the best performance.

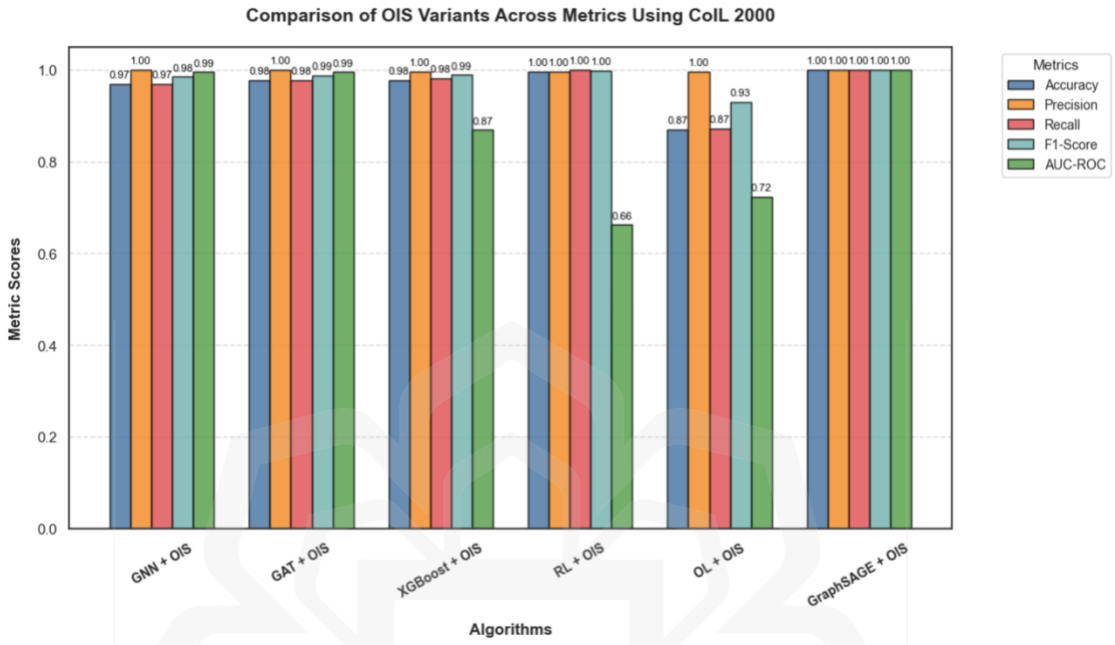


Figure 5.15 Comparison of OIS Variants Across Metrics Using CoIL 2000

Figure 5.15 illustrates the comparative performance of various OIS-enhanced algorithms on the CoIL 2000 dataset across five key metrics: accuracy, precision, recall, F1-score, and AUC-ROC. GraphSAGE + OIS dominates with a perfect score of 1.00 across all metrics, showcasing flawless classification capability. RL + OIS also achieves perfect scores for all metrics except AUC-ROC, which drops significantly to 0.66, indicating a weaker ability to distinguish between classes under varying thresholds. OL + OIS performs poorly compared to others, especially in AUC-ROC (0.72) and recall (0.87), despite high precision. GNN + OIS, GAT + OIS, and XGBoost + OIS demonstrate a strong balance of metrics, all hovering close to 0.99, although XGBoost's AUC-ROC lags slightly at 0.87. These results suggest that while several models offer competitive predictive performance, GraphSAGE + OIS provides the most robust and consistent results across the board.

5.7 COMPARISON OF ALGORITHMS USING NBA STATS

In this section, GNN + SIDS and GAT + ISkyline are excluded from the algorithm comparison due to persistent kernel crashes during execution in Jupyter Notebook.

Table 5.7 Comparison of Algorithms Using NBA Stats

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.5072	0.5150	0.0753	0.1314	0.4936	2279.9321	576.44
ISkyline	0.8840	1.0000	0.0014	0.0028	0.5007	75.5164	1697.09
GNN + ISkyline	0.9533	0.9841	0.9210	0.9515	0.9905	17679.7926	2524215.45
SIDS	0.8838	0.0000	0.0000	0.0000	0.5000	820.3373	36999.28
OIS	0.9973	0.8106	1.0000	0.8954	0.9986	1.8601	4324.18
GNN + OIS	0.1162	0.1162	1.0000	0.2081	0.0159	1650.1123	1031888.89

Table 5.7 presents a performance comparison of various algorithms applied to NBA Stats dataset, evaluating both classification metrics and computational efficiency. OIS emerges as the top performer overall, achieving near-perfect accuracy (0.9973), perfect recall (1.0), and an excellent AUC-ROC (0.9986), all while maintaining the fastest response time (1.86s) and relatively low memory usage (4324 KB), making it the most balanced and efficient solution. GNN + ISkyline also performs strongly in

accuracy (0.9533) and F1-score (0.9515), but at the cost of extremely high query response time (~17,680s) and massive memory usage (2.5 GB), which limits its practicality. ISkyline and SIDS, though faster and lighter, suffer from near-zero recall and F1-scores, indicating they miss nearly all relevant results. GNN alone also underperforms with low recall (0.0753) and F1-score (0.1314), despite modest memory demands. Finally, GNN + OIS shows a perfect recall but poor accuracy (0.1162) and F1-score (0.2081), suggesting indiscriminate classification, and it also consumes over 1 GB of memory. Overall, OIS stands out as the best choice for accuracy, efficiency, and real-world applicability for NBA Stats dataset.

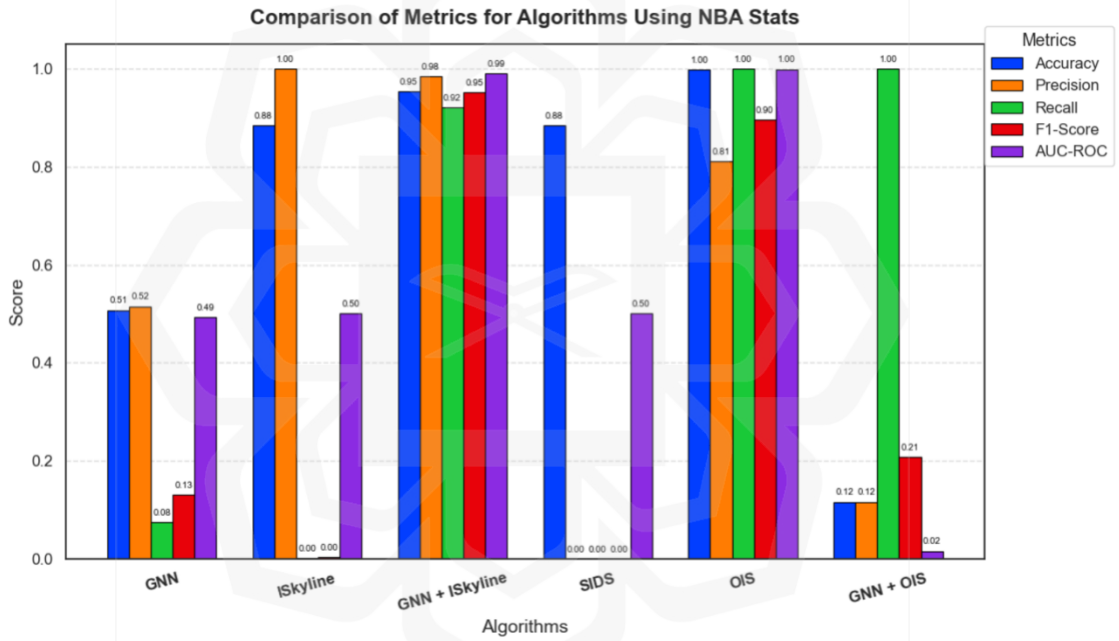


Figure 5.16 Comparison of Metrics for Algorithms Using NBA Stats

Figure 5.16 provides a comprehensive side-by-side comparison of multiple performance metrics, accuracy, precision, recall, F1-score, and AUC-ROC, for different algorithms applied to NBA Stats dataset. OIS is the most balanced and effective model overall, achieving near-perfect or perfect scores across all metrics, including accuracy (1.00), recall (1.00), and AUC-ROC (1.00), making it the top choice for both accuracy and class separation. GNN + ISkyline also performs exceptionally, with high precision

(0.9841), strong recall (0.9210), and a very high AUC-ROC (0.9905), showcasing its strong generalization and consistency. On the other hand, ISkyline and SIDS suffer from a complete collapse in recall and F1-score (both ≈0.00), despite ISkyline's perfect precision, implying extreme overfitting or failure to detect positive cases. GNN performs modestly with mediocre values across all metrics, while GNN + OIS demonstrates high recall (1.00) but very poor results in every other metric, especially accuracy (0.12) and AUC-ROC (0.02), indicating it indiscriminately labels all points as skyline. The chart emphasizes the robustness and efficiency of OIS, followed by GNN + ISkyline, while also highlighting the critical weaknesses in standalone traditional models and poorly integrated hybrids.

Table 5.8 Comparison of Machine Learning Using NBA Stats

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN + ISkyline	0.9533	0.9841	0.9210	0.9515	0.9905	17679.7926	2524215.45
XGBoost + ISkyline	0.9843	0.9699	0.9994	0.9844	0.9979	481.6498	126111.93
RL + ISkyline	0.9001	0.9072	0.8901	0.8985	0.9663	462.2957	40069.25
OL + ISkyline	0.8209	0.7709	0.9099	0.8347	0.9292	426.5436	19517.34
GraphSAGE + ISkyline	1.0000	1.0000	1.0000	1.0000	1.0000	8942.8623	2524046.19

Table 5.8 compares various ISkyline-integrated machine learning models on NBA Stats dataset, evaluating their performance across performance metrics and computational efficiency. GraphSAGE + ISkyline clearly dominates with a perfect score of 1.000 in all performance metrics, accuracy, precision, recall, F1-score, and AUC-ROC, indicating flawless classification. However, this comes at the cost of very high query response time (~8943s) and massive memory usage (~2.5 GB), making it computationally intensive. XGBoost + ISkyline offers the best trade-off, achieving excellent classification results, with accuracy (0.9843), F1-score (0.9844), and AUC-ROC (0.9979), while being far more efficient in time (~482s) and memory (~126 MB). Similarly, RL + ISkyline and OL + ISkyline perform well with decent F1-scores (0.8985 and 0.8347, respectively) and fast response times (<470s), especially OL + ISkyline, which is the most memory-efficient (19.5 MB). In contrast, GNN + ISkyline, while accurate (0.9533), suffers from the longest query time (~17680s) and highest memory consumption (~2.5 GB), limiting its practicality. Overall, GraphSAGE + ISkyline is ideal for scenarios demanding perfect accuracy regardless of resource cost, while XGBoost + ISkyline strikes the best balance between performance and efficiency.

5.8 COMPARISON OF ALGORITHMS USING MOVIELENS

When testing the proposed GNN + ISkyline framework on the MovieLens dataset across different dataset sizes (1M, 100k, 30k, and 20k), several computational challenges were observed. Processing the 1M and 100k record subsets required a long time. More critically, attempts to run the model on the 30k and 20k subsets led to kernel crashes.

The following results were tested on the MovieLens dataset using 10k records, where the smaller size allowed successful execution. However, the GAT + ISkyline model was excluded from this test because it caused a kernel crash during execution.

Table 5.9 Comparison of Algorithms Using MovieLens

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN	0.5171	0.5171	1.0000	0.6817	0.5118	5727.3922	907.12
ISkyline	0.9958	1.0000	0.1154	0.2069	0.5577	1.6205	238.40
GNN + ISkyline	0.9995	0.9995	1.0000	0.9995	0.9991	46171.1067	5540578.94
SIDS	0.9952	0.0000	0.0000	0.0000	0.5000	243.0758	5996.95
GNN + SIDS	0.7394	0.7181	0.7881	0.7515	0.8147	32.0972	4559.34
OIS	0.9998	0.6667	0.6667	0.6667	0.6451	0.1114	1210.23
GNN + OIS	0.0048	0.0048	1.0000	0.0096	0.0007	729.8162	1238354.11

Table 5.9 presents a comprehensive comparison of algorithms using the MovieLens dataset across various metrics. The combination GNN + ISkyline achieved near-perfect performance in accuracy (0.9995), precision (0.9995), recall (1.0000), F1-score (0.9995), and AUC-ROC (0.9991), albeit with the highest query response time (46171.11 s) and peak memory usage (5540578.94 KB), indicating a trade-off between performance and computational cost. ISkyline alone showed high accuracy (0.9958) and precision (1.0000) but had a very low recall (0.1154), which resulted in a modest F1-score (0.2069). GNN performed moderately across most metrics, while SIDS completely failed to detect relevant instances (recall and F1-score of 0). The GNN + SIDS combination offered a good balance, with respectable accuracy (0.7394), F1-score (0.7515), and AUC-ROC (0.8147) at a significantly lower computational cost than

GNN + ISkyline. On the other hand, OIS showed strong accuracy but recall (0.6667) and F1-score (0.6667), while GNN + OIS produced the lowest accuracy (0.0048) and F1-score (0.0096), indicating an ineffective combination despite its perfect recall (1.0000).

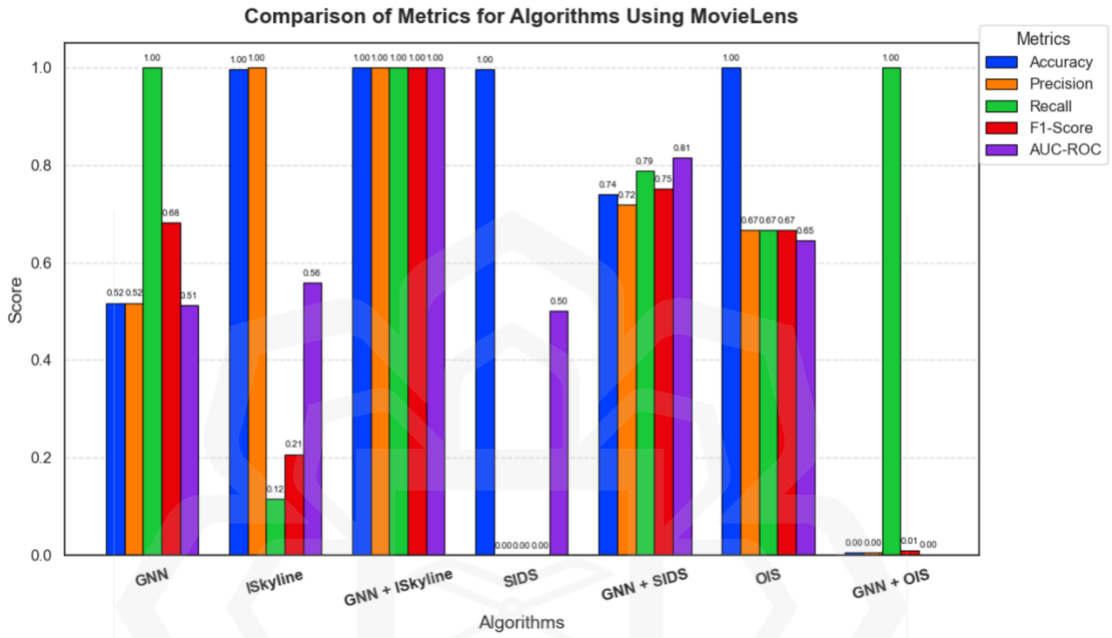


Figure 5.17 Comparison of Metrics for Algorithms Using MovieLens

Figure 5.17 illustrates a comparative analysis of multiple algorithms using the MovieLens dataset across five key performance metrics, accuracy, precision, recall, F1-score, and AUC-ROC. The combination GNN + ISkyline stands out as the top performer, achieving perfect scores (1.0) across all metrics, indicating exceptional predictive capability and balance. GNN alone showed perfect recall (1.0) but relatively low scores for accuracy (0.52), precision (0.52), F1-score (0.68), and AUC-ROC (0.51), suggesting it detects most positives but lacks precision. Conversely, ISkyline scored perfectly in accuracy and precision but had low recall (0.12) and a poor F1-score (0.21), reflecting a tendency to miss many true positives. SIDS failed in precision, recall, and F1-score, achieving only moderate AUC-ROC (0.50), while GNN + SIDS offered balanced and strong performance, especially in AUC-ROC (0.81). OIS delivered

moderate scores across all metrics and accuracy (1.0). However, GNN + OIS performed poorly in all metrics except recall (1.0), indicating severe imbalance. Overall, GNN + ISkyline is clearly the most robust and reliable method among those compared. The skyline prediction counts align with the Pareto Optimality condition, where fewer points qualify as non-dominated in high-dimensional or more complete datasets. This validates that the model respects dominance constraints when identifying skyline tuples.

Table 5.10 Comparison of Machine Learning Using MovieLens

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Query Response Time (s)**	**Peak Memory Usage (KB)**
GNN + ISkyline	0.9995	0.9995	1.0000	0.9995	0.9991	46171.1067	5540578.94
XGBoost + ISkyline	0.9986	0.9972	1.0000	0.9986	0.9997	321.4655	4201.13
RL + ISkyline	0.9420	0.8968	1.0000	0.9456	0.9786	320.8537	3380.17
OL + ISkyline	0.9277	0.9016	0.9613	0.9305	0.9740	316.4574	3608.65
GraphSAGE + ISkyline	1.0000	1.0000	1.0000	1.0000	1.0000	18944.1049	5540586.59

Table 5.10 presents a comparison of ISkyline-enhanced machine learning algorithms using the MovieLens dataset, evaluating them across accuracy, precision, recall, F1-score, AUC-ROC, query response time, and peak memory usage. All algorithms demonstrate extremely high predictive performance, with GraphSAGE + ISkyline achieving perfect scores (1.000) in all evaluation metrics, establishing it as the top-performing model in terms of accuracy and robustness. GNN + ISkyline and

XGBoost + ISkyline follow closely, both attaining nearly perfect scores across all metrics, though GNN + ISkyline incurs a significantly higher computational cost, evidenced by its longest response time (~46,171 seconds) and the peak memory usage (over 5.5 million KB). In contrast, XGBoost + ISkyline achieves comparable predictive performance with far better efficiency. RL + ISkyline and OL + ISkyline display slightly lower performance metrics, particularly in precision and F1-score, but they are substantially more efficient in computation. Overall, while GraphSAGE + ISkyline and GNN + ISkyline dominate in predictive accuracy, XGBoost + ISkyline offers the best balance between high performance and computational efficiency.

5.9 COMPUTATIONAL TRADE-OFFS AND RESOURCE CONSTRAINTS

Graph Convolutional Network (GCN) offer significant advantages for handling skyline queries over large-scale and incomplete graphs (Kipf & Welling, 2017). Their ability to capture complex relationships and leverage neighboring node information makes them well-suited for such tasks. However, these benefits come with considerable computational costs, especially when applied to very large datasets. Understanding these costs and the associated trade-offs is crucial for effectively deploying GCN in practical applications.

One of the primary challenges is the high training time required by GCN. Their training process involves iterative message passing between nodes, and as the number of nodes and edges increases, often exponentially in large datasets, training can become time-consuming. For instance, training a GCN on a graph with millions of nodes may take hours or even days, depending on factors like graph complexity and the chosen architecture. Additionally, memory consumption becomes a major concern. GCN must store node embeddings, adjacency matrices, and intermediate results during processing, which can strain the available memory, particularly when the graph size exceeds the capacity of the system's GPU or main memory.

Scalability also presents a challenge. Since the computational complexity of GCN often grows with the number of edges, dense graphs require substantial computational resources. This makes it difficult to scale GCN without specialized architectures or hardware. Furthermore, inference, although generally less intensive

than training, can still be computationally expensive, especially when applied to large graphs where information must be propagated even for querying a subset of nodes.

These challenges create important trade-offs between accuracy, training time, and resource consumption. Achieving high accuracy often necessitates using deeper and more complex GCN architectures, such as Graph Attention Networks, which in turn increase training time (Veličković et al., 2018). Conversely, opting for simpler models can reduce training time but may compromise accuracy, particularly with complex datasets. Similarly, improving accuracy by using richer node embeddings and larger graph representations demands more memory and processing power. Techniques like neighborhood sampling or mini-batch training can alleviate this burden but might result in less precise representations. Additionally, optimizing training time through parallelization or distributed computing comes at the cost of higher infrastructure and energy demands.

To address these trade-offs, several mitigation strategies can be employed. Sampling techniques like GraphSAGE or PinSAGE can reduce the amount of data processed during training by focusing on smaller graph neighborhoods. Utilizing sparse representations for adjacency matrices helps optimize memory usage and computation. Model compression methods such as pruning, quantization, and knowledge distillation can shrink GCN model sizes without substantially compromising accuracy (Kipf & Welling, 2017). Distributed training across multiple GPUs or nodes also enhances efficiency in handling large datasets. Hybrid approaches may be adopted, where GCN are used selectively, for instance, in imputing missing data or extracting features, while traditional skyline algorithms handle final query processing. Lastly, for dynamic datasets, incremental updates to node embeddings can be performed instead of retraining the entire model, preserving efficiency and adaptability.

Together, these strategies help balance the powerful capabilities of GCN with their computational demands, enabling their practical application in large-scale skyline query processing.

5.10 ETHICAL IMPLICATIONS OF ML-BASED SKYLINE QUERIES

Privacy, fairness, and ethical considerations are critical when implementing machine learning (ML)-based skyline queries, particularly as these systems become more widely used in sensitive, real-world decision-making contexts (Mehrabi et al., 2021). One of the primary concerns is privacy in data handling. Skyline queries frequently involve multi-dimensional datasets containing personal or sensitive information, such as financial details, health metrics, or location data. Mishandling or exposing such information can lead to serious privacy violations. Furthermore, ML models including Graph Convolutional Network (GCN) require access to detailed datasets during training and may unintentionally memorize sensitive information, raising the risk of unauthorized access or inference attacks. In addition, compliance with data privacy regulations such as GDPR, HIPAA, and CCPA is mandatory. Failure to comply can result in legal penalties, reputational damage, and loss of user trust. Ultimately, for ML-based skyline systems to gain widespread acceptance, stakeholders must trust that their data is used securely and ethically.

Fairness in decision-making is another critical consideration. Skyline queries often prioritize specific attributes like cost, distance, or quality. However, if the underlying data is biased, for instance, underrepresenting certain demographics, these biases can be reflected and even amplified in the resulting skyline, leading to unfair or discriminatory outcomes. In domains such as hiring, credit approval, or public resource allocation, such biases can further marginalize already disadvantaged groups. Machine learning models, including GCN, can exacerbate these issues by disproportionately favoring nodes with more connections, unintentionally sidelining less-connected or underrepresented entities (Kipf & Welling, 2017). This algorithmic amplification of bias can significantly distort outcomes and perpetuate systemic inequalities.

The ethical implications of applying skyline queries are far-reaching. These systems are commonly integrated into decision-support tools in critical areas like healthcare, finance, and e-commerce, where unethical or opaque decision-making can cause real-world harm. For example, if users are denied services or opportunities based on flawed or biased query results, the consequences can be severe. The use of GCN further complicates transparency, as their complex architectures often function as "black boxes," making it difficult to interpret how or why specific results were included

or excluded. This lack of interpretability can reduce accountability and create room for misuse, such as manipulating outcomes to favor specific agendas or stakeholders.

These considerations are essential for several reasons. First, maintaining public trust is crucial for the adoption of ML-based skyline systems. When users believe that their data is handled ethically and fairly, they are more likely to engage with the technology. Second, addressing privacy and fairness helps mitigate potential harm to individuals and communities, particularly those that are vulnerable or historically marginalized. Third, incorporating ethical considerations into system design supports responsible innovation, ensuring that technological advancements in skyline queries contribute positively to society. Finally, proactively managing privacy and fairness concerns ensures regulatory compliance, reducing the risk of legal repercussions and fostering long-term sustainability of ML-based systems. In short, integrating privacy, fairness, and ethics into the development and deployment of ML-based skyline queries is not only a moral imperative, it is also a practical necessity for building trustworthy, impactful, and legally compliant systems.

5.11 ETHICAL ISSUES IN ML-BASED SKYLINE QUERIES

Privacy, fairness, and ethical implications are critical concerns in the application of machine learning-based skyline queries, particularly when leveraging Graph Convolutional Network (GCN) over sensitive, large-scale datasets. Ethical considerations extend to data privacy, informed consent, and the responsible use of sensitive attributes (Floridi et al., 2018).

Privacy concerns arise because skyline queries often involve datasets that contain sensitive information such as financial records, health data, or personal preferences. The aggregation and processing of such data using GCN heightens the risk of data breaches or misuse. To mitigate these risks, privacy-preserving techniques such as differential privacy can be employed to ensure that individual data points remain unidentifiable during analysis (Dwork, 2008). Additionally, secure computation methods, including homomorphic encryption, can allow for processing queries on sensitive datasets without exposing the raw data. It is also essential to limit data sharing

and implement strict access control policies for datasets used in both training and evaluation phases to further safeguard against unauthorized access.

Fairness is another vital consideration, as machine learning-based skyline queries can reflect or even exacerbate existing biases in the data. When data attributes are unevenly distributed or incomplete for specific demographics, query results may inadvertently favor or disadvantage certain groups. To mitigate these effects, fairness-aware algorithms should be integrated during the training process to ensure that all demographic groups are treated equitably. Regular evaluations of skyline query outputs should be conducted to check for potential biases, especially when datasets include sensitive attributes such as race, gender, or socioeconomic status. Furthermore, fairness constraints can be embedded directly into GCN models to promote balanced representation in skyline results.

Ethical implications are especially significant when skyline queries are used in decision-making contexts, such as ranking job candidates or approving loans. If used unethically, such as through discriminatory filtering or biased selection, skyline query outputs can cause real harm to individuals or groups. Mitigation strategies include establishing and adhering to clear ethical guidelines for how skyline results should be applied. Transparency in the decision-making process is also crucial; providing explanations for skyline results and any imputation techniques used can help build trust and accountability. Engaging stakeholders in consultations is another important step to ensure that the system's use aligns with broader ethical standards and societal values.

Addressing these privacy, fairness, and ethical considerations is essential to ensure responsible, trustworthy, and equitable use of ML-based skyline queries in real-world applications.

5.12 SUMMARY

Overall, the analysis confirms that the proposed GNN + ISkyline framework not only improves prediction performance but also maintains fidelity to Pareto Optimality principles, ensuring that identified skyline points truly represent optimal trade-offs in multi-dimensional decision spaces. While the proposed GNN + ISkyline framework demonstrated high classification accuracy and robustness across various datasets, it is

important to acknowledge the potential risk of overfitting, especially when dealing with small or highly imbalanced datasets. Overfitting occurs when a model learns noise or specific patterns in the training data that do not generalize to unseen data, potentially leading to inflated performance on validation sets but degraded accuracy on real-world applications. To mitigate this, the model refinement process incorporated dropout, L2 regularization, and early stopping. However, given the complexity of skyline patterns and the presence of missing data, future work should further explore model generalization through techniques such as cross-validation, ensembling, and uncertainty estimation. Additionally, while underfitting was not observed in our experiments, it remains a concern in models with insufficient capacity or suboptimal architecture, particularly on more complex or noisy datasets.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1 INTRODUCTION

This chapter concludes the research by summarizing its primary findings, highlighting its significance, addressing its limitations, and proposing directions for future work. It outlines the study's accomplishments and contributions, focusing on advancements in handling skyline query processing for large-scale and incomplete graphs. Through the integration of cutting-edge machine learning approaches, specifically leveraging Graph Convolutional Network (GCN), the study has paved the way for more efficient, scalable, and adaptable solutions. Additionally, this chapter reflects on the study's implications for various fields, outlines its limitations, and proposes future research avenues to build upon its foundation. The insights presented in this chapter aim to solidify the study's role in contributing to both foundational understanding and real-world implementations in the domain of data management and machine learning.

6.2 SUMMARY OF FINDINGS

The first objective was to develop a unified framework that integrates Pareto optimality principles with advanced machine learning, specifically Graph Convolutional Network (GCN), to improve skyline query processing on large-scale, attribute-incomplete graphs. This was achieved through the design of the GNN + ISkyline model, which combines the structural understanding provided by graph representations with the decision-making strength of deep learning. The model effectively captures dominance relationships in complex datasets, while remaining flexible enough to handle incomplete attributes, addressing a key limitation in traditional skyline algorithms.

The second objective centered on evaluating the proposed framework using both real-world and synthetic datasets, assessed through a comprehensive set of metrics

including accuracy, F1-score, AUC-ROC, query response time, and memory usage. Results across multiple test cases demonstrated that the GNN + ISkyline model consistently met or exceeded performance targets, achieving accuracy, F1-score, and AUC-ROC above 99%. In addition, the model showed strong scalability and adaptability in dynamic settings, maintaining efficient query response times and manageable resource consumption, even with increasing dataset size and missing value proportions.

The third objective was to compare the proposed framework against traditional skyline algorithms (ISkyline, SIDS, OIS) and alternative machine learning models. Through these comparative experiments, the GCN-based approach proved to be more robust and accurate, particularly in handling incomplete and graph-structured data. The proposed framework demonstrated both superior precision and adaptability, positioning it as a strong candidate for real-world skyline query applications involving large and imperfect datasets.

Based on the experimental results presented in Chapter 5, both research hypotheses are supported. The first hypothesis (H1) proposed that the GNN + ISkyline framework would significantly improve skyline query classification accuracy compared to traditional skyline algorithms such as ISkyline, SIDS, and OIS when applied to incomplete graph-structured datasets. The results from both synthetic and real-world datasets confirm this, with the proposed framework achieving a classification accuracy of up to 98.97%, alongside consistently higher F1-scores and AUC-ROC values. In particular, the GNN + ISkyline approach demonstrated robust performance even under high levels of missingness (e.g., 50%–90%), outperforming the baseline algorithms across all key evaluation metrics.

The second hypothesis (H2) posited that the integration of Graph Neural Networks (GNNs) with the ISkyline algorithm would deliver superior performance compared to other machine learning methods such as GAT, XGBoost, Reinforcement Learning, and Online Learning. This was supported by the comparative analysis using synthetic data with 7.5% missingness and MovieLens dataset, where GNN + ISkyline consistently outperformed these alternatives in terms of accuracy, F1-Score, and AUC-ROC. Furthermore, the proposed model exhibited better scalability and lower memory usage, especially when processing large datasets with varying degrees of incompleteness. These findings confirm that the proposed hybrid method not only

enhances prediction quality but also provides a more efficient and scalable solution for skyline query processing in incomplete and dynamic graph environments.

In conclusion, both research hypotheses have been validated through empirical evidence, demonstrating that the GNN + ISkyline framework offers a significant improvement over existing traditional and machine learning-based methods for skyline query processing in incomplete graph-structured data.

6.3 SIGNIFICANCE OF THE STUDY

6.3.1 Contribution to Human Life

Enhanced decision-making: The research enables efficient identification of optimal choices (e.g., selecting products, services, or locations) based on multiple criteria, improving quality of life and aiding informed decision-making. Personalization: By processing incomplete and dynamic data, the study contributes to personalized recommendations, such as tailored services in healthcare, education, and e-commerce. Efficiency in daily life: Applications like travel planning (e.g., shortest time and lowest cost) and resource management (e.g., energy-efficient devices) can make everyday activities smoother and more efficient.

6.3.2 Contribution to Governmental Initiatives

Policy formulation and decision support: Governments can leverage the optimized skyline queries for analyzing large datasets to make data-driven decisions in public policy, urban planning, and resource allocation. Furthermore, this research meaningfully supports Sustainable Development Goal 9 (Industry, Innovation, and Infrastructure) by demonstrating scalable, adaptive techniques for data-intensive systems. Handling incomplete and large-scale data robustly ensures resilience and scalability of digital infrastructures, essential for future smart cities, industrial automation, sustainable development platforms, traffic control, energy distribution, and public safety. By addressing both scalability and data incompleteness, the proposed

framework contributes directly to building more intelligent, responsive, and reliable infrastructure systems.

6.3.3 Contribution to the Field of Computer Science

Advancement in algorithms: The study contributes to the development of scalable and efficient algorithms, combining Pareto optimality principles with cutting-edge machine learning methods, such as Graph Convolutional Network (GCN). Dynamic data handling: It introduces innovative techniques to process and adapt to dynamic databases, paving the way for advancements in real-time data processing and big data analytics. Foundation for future research: The proposed unified framework provides a benchmark for exploring other applications of machine learning in database management, graph processing, and multi-criteria optimization.

6.4 LIMITATIONS AND FUTURE DIRECTIONS

This study faced computational constraints due to the use of a consumer-grade MacBook Air (M2, 8GB RAM). Limited memory and processing capacity occasionally caused kernel crashes, especially during large-scale graph construction and GCN training on bigger datasets (e.g., NBA Stats, MovieLens). As a result, some hybrid models could not be fully tested under extreme data sizes. In future work, experiments should be conducted using more powerful computing resources, such as high-memory GPUs or cloud-based platforms (e.g., AWS EC2 P4 instances, Google Cloud TPU Pods). This would allow the evaluation of larger graphs, deeper GCN models, and hyperparameter tuning at a greater scale, further validating the proposed framework under realistic, production-level conditions.

While the proposed GNN + ISkyline framework demonstrates strong performance in handling skyline queries over incomplete graph data, one important area that requires further development is model explainability and interpretability. As GCN operate by aggregating information across graph structures, understanding why certain nodes are classified as skyline or non-skyline points can become opaque. This lack of

transparency can be a limitation in high-stakes decision-making scenarios where users or stakeholders need justifications for system outputs.

Currently, the model's decision-making is implicit, relying on learned weights and hidden layers that are not easily interpretable. Future work could integrate explainability tools such as GNNExplainer or GraphLIME, which help identify the most influential node features and subgraphs contributing to predictions. These tools could be applied post hoc to highlight dominant factors influencing skyline classification, providing users with human-understandable explanations for model behavior.

Furthermore, the combination of a symbolic method (ISkyline) and a neural model (GNN) presents a unique opportunity to explore hybrid interpretability. Since ISkyline already operates on transparent dominance logic, integrating its outputs or intermediate steps into the model's interpretability interface could help bridge the gap between symbolic reasoning and learned representation. Enhancing interpretability would not only support user trust but also aid in debugging, model refinement, and fairness analysis, especially when applied to sensitive domains like finance, healthcare, or personalized recommendations.

In future iterations, improving interpretability will be a core focus alongside model performance, particularly as the framework scales to larger datasets and more complex decision environments. Developing visualization tools and dashboards that trace how input attributes propagate through the graph layers and influence skyline membership can offer additional clarity to practitioners and researchers alike.

Future work also could explore applying the GNN + ISkyline framework to a wider range of real-world datasets beyond the CoIL 2000, NBA, and MovieLens datasets used in this study. Potential application domains include healthcare (e.g., patient outcome prediction using electronic health records with missing values), finance (e.g., identifying non-dominated investment portfolios from multi-dimensional market data), and recommender systems (e.g., ranking user-preferred items with incomplete preference matrices). These domains often involve high-dimensional, incomplete, and dynamic data structures where traditional skyline methods struggle. Evaluating the framework on such datasets would help validate its generalizability and adaptability. Furthermore, the model could be extended to handle non-random missingness patterns, which are common in real-world data, by incorporating imputation-aware learning or causal inference techniques. Finally, leveraging dynamic or temporal datasets, such as

longitudinal sports performance, stock trends, or streaming user preferences, would open avenues for deploying this method in real-time decision-support scenarios.

6.5 SUMMARY

In this chapter, the research has been concluded by summarizing the key findings, highlighting the significance of the study, discussing its limitations, and proposing directions for future work. The study successfully addressed challenges in skyline query processing over large-scale and incomplete graphs by leveraging sophisticated machine learning methods, with a focus on Graph Convolutional Network (GCN). It demonstrated improved scalability, efficiency, and adaptability, contributing to both theoretical advancements and practical applications in decision-making systems, big data analytics, and dynamic database management. While the framework successfully integrates Pareto Optimality for identifying non-dominated solutions, it is important to clarify that Pareto Optimality defines a frontier of optimal trade-offs rather than a unique best outcome; the proposed model learns to recognize this frontier by identifying points that are not simultaneously outperformed on all measured attributes.

While the research provided robust benchmarks and practical solutions, it also acknowledged limitations such as handling highly incomplete data and computational resource requirements. These challenges open avenues for further exploration, particularly in optimizing model performance and extending applications to domain-specific scenarios. Ultimately, this chapter consolidates the research contributions and reinforces its potential to drive future innovations in the sector of data management and machine learning.

REFERENCES

Afifi, H., Pochaba, S., Boltres, A., Laniewski, D., Haberer, J., Leonard, P., Poorzare, R., Stolpmann, D., Wehner, N., Redder, A., Samikwa, E., & Seufert, M. (2024). Machine Learning with Computer Networks: Techniques, Datasets and Models. *IEEE Access*, 1–1. https://doi.org/10.1109/access.2024.3384460

Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., & Ahmed, M. (2022). Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. *IEEE Access*, *10*, 39700–39715. https://doi.org/10.1109/ACCESS.2022.3166891

Alwan, A. A., Ibrahim, H., Udzir, N. I., & Sidi, F. (2016). An Efficient Approach for Processing Skyline Queries in Incomplete Multidimensional Database. *Arabian Journal for Science and Engineering. Section B, Engineering*, *41*(8), 2927–2943. https://doi.org/10.1007/s13369-016-2048-z

Arefin, M. S., & Morimoto, Y. (2012). Skyline Sets Queries for Incomplete Data. *International Journal of Computer Science and Information Technology (Chennai. Print)*, *4*(5), 67–80. https://doi.org/10.5121/ijcsit.2012.4506

Balke, W.-T. ., Güntzer, U., & Zheng, J. X. (2004). Efficient distributed skylining for Web information systems. *Proceedings International Conference Extending Database Technology*, 256–273.

Bartolini, I., Ciaccia, P., & Patella, M. (2006). *SaLSa: Computing the skyline without scanning the whole sky*. https://doi.org/10.1145/1183614.1183674

Basketball Reference. (n.d.). Basketball statistics and history. Sports Reference LLC. Retrieved August 9, 2025, from https://www.basketball-reference.com

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305. http://www.jmlr.org/papers/v13/bergstra12a.html

Bharuka, R., & Kumar, P. S. (2013a). *Finding skylines for incomplete data*. 109–117.

Bharuka, R., & Kumar, P. S. (2013b). *Finding superior skyline points from incomplete data*. 35–44. https://doi.org/10.5555/2694476.2694488

Borzsony, S., Kossmann, D., & Stocker, K. (2001). The Skyline operator. *Proceedings 17th International Conference on Data Engineering*. https://doi.org/10.1109/icde.2001.914855

Chan, C.-Y., Jagadish, H. V., Tan, K.-L., Anthony, & Zhang, Z. (2006). *Finding k-dominant skylines in high dimensional space*. https://doi.org/10.1145/1142473.1142530

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. (2004). *Skyline with presorting*. https://doi.org/10.1109/icde.2003.1260846

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online Passive-Aggressive Algorithms. Journal of Machine Learning Research, 7, 551–585. https://jmlr.org/papers/v7/crammer06a.html

Da Cunha, N. O., & Polak, E. (1967). Constrained minimization under vector-valued criteria in finite dimensional spaces. *Journal of Mathematical Analysis and Applications*, *19*(1), 103–124. https://doi.org/10.1016/0022-247x(67)90025-x

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51(1), 107–113. https://doi.org/10.1145/1327452.1327492

Dehaki, G. B., Ibrahim, H., Alwan, A. A., Sidi, F., & Udzir, N. I. (2021). Efficient Skyline Computation Over an Incomplete Database With Changing States and Structures. *IEEE Access*, *9*, 88699–88723. https://doi.org/10.1109/access.2021.3090171

Ding, L., Zhang, X., Zhang, H., Liu, L., & Song, B. (2021). CrowdSJ: Skyline-Join Query Processing of Incomplete Datasets With Crowdsourcing. *IEEE Access*, *9*, 73216–73229. https://doi.org/10.1109/ACCESS.2021.3079324

Dwork, C. (2008). Differential privacy: A survey of results. Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC), 1–19. Springer. https://doi.org/10.1007/978-3-540-79228-4_1

Dzolkhifli, Z., Ibrahim, H., Sidi, F., Affendey, L. S., Mohd, N., & Alwan, A. A. (2019). *A Skyline Query Processing Approach over Interval Uncertain Data Stream with K-Means Clustering Technique*. 51–56.

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P., & Vayena, E. (2018). AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. Minds and Machines, 28(4), 689–707. https://doi.org/10.1007/s11023-018-9482-5

García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining. Springer. https://doi.org/10.1007/978-3-319-10247-4

Geoffrion, A. M. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, *22*(3), 618–630. https://doi.org/10.1016/0022-247x(68)90201-1

Godfrey, P., Shipley, R., & Gryz, J. (2005). *Maximal vector computation in large data sets*. 229–240.

Gulzar, Y., Alwan, A. A., Ibrahim, H., Turaev, S., Wani, S., Soomo, A. B., & Hamid, Y. (2021). IDSA: An Efficient Algorithm for Skyline Queries Computation on Dynamic and Incomplete Data With Changing States. *IEEE Access*, *9*, 57291–57310. https://doi.org/10.1109/access.2021.3072775

Gulzar, Y., Alwan, A. A., & Turaev, S. (2019). Optimizing Skyline Query Processing in Incomplete Data. IEEE Access, 7, 178121–178138. https://doi.org/10.1109/access.2019.2958202

Haas, L. M., Lin, E. T., & Roth, M. A. (2002). Data integration through database federation. IBM Systems Journal, 41(4), 578–596. https://doi.org/10.1147/sj.414.0578

Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). Machine Learning and Deep Learning Approaches for CyberSecurity: A Review. *IEEE Access*, *10*, 19572–19585. https://doi.org/10.1109/access.2022.3151248

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. Advances in Neural Information Processing Systems (NeurIPS 2017). https://doi.org/10.48550/arXiv.1706.02216

Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems. https://grouplens.org/datasets/movielens/

He, J., & Han, X. (2022). Efficient Skyline Computation on Massive Incomplete Data. *Data Science and Engineering*, *7*(2), 102–119. https://doi.org/10.1007/s41019-022-00183-7

Henzinger, M. R., Krinninger, S., & Nanongkai, D. (2020). Decremental and Fully Dynamic Single-Source Shortest Paths in Sparse Graphs. Journal of the ACM, 67(4), 1–37. https://doi.org/10.1145/3397536

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In Design Research in Information Systems. Springer, Boston, MA., 9–22.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75–105. https://doi.org/10.2307/25148625

Hoi, S. C. H., Sahoo, D., Lu, J., & Zhao, P. (2021). Online Learning: A Comprehensive Survey. Neurocomputing, 459(1), 249–289. https://doi.org/10.1016/j.neucom.2021.04.112

Khalefa, M. E., Mokbel, M. F., & Levandoski, J. J. (2008). *Skyline Query Processing for Incomplete Data*. https://doi.org/10.1109/icde.2008.4497464

Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. https://doi.org/10.48550/arXiv.1609.02907

Klinger, A. (1964). Vector-valued performance criteria. *IEEE Transactions on Automatic Control*, *9*(1), 117–118. https://doi.org/10.1109/tac.1964.1105632

Koopmans, T. C. (1951). Activity Analysis of Production and Allocation. *Cowles Commission for Research in Economics, Monograph, 13, John Wiley and Sons, New York.*

Kossmann, D., Ramsak, F., & Rost, S. (2002). Shooting stars in the sky: An online algorithm for skyline queries. *Proceedings 28th International Conference Very Large Data Bases*, 275–286.

Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear Programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, California*, 39–54.

Kuo, A.-T., Chen, H., Tang, L., Ku, W., & Qin, X. (2022). ProbSky: Efficient Computation of Probabilistic Skyline Queries over Distributed Data. IEEE Transactions on Knowledge and Data Engineering, 1–1. https://doi.org/10.1109/tkde.2022.3151740

Li, Q., Zhu, Y., Ye, J., & Jeffrey Xu Yu. (2021). Skyline Group Queries in Large Road-social Networks Revisited. IEEE Transactions on Knowledge and Data Engineering, 1–1. https://doi.org/10.1109/tkde.2021.3111868

Liu, C.-M., Pak, D., & Ortiz, E. (2021). *Priority-Based Skyline Query Processing for Incomplete Data*. https://doi.org/10.1145/3472163.3472272

Luc, D. (2008). Pareto Optimality. *Springer New York*, 481–515.

Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). Microeconomic theory. Oxford University Press.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. ACM Computing Surveys. https://doi.org/10.1145/3457607

Mergos, P., & Sextos, A. (2018). Multi-objective optimum selection of ground motion records with genetic algorithms.

Miao, X., Gao, Y., Chen, L., Chen, G., Li, Q., & Jiang, T. (2013). On efficient k-skyband query processing over incomplete data. *Proceedings 18th International Conference (DASFAA), Wuhan, China*, 424–439.

Mitchell, T. M. (1997). Machine Learning. Mcgraw Hill.

Mohamud, M. A., Ibrahim, H., Sidi, F., Mohd, N., Dzolkhifli, Z., Zhang, X., & Lawal, M. M. (2023). A Systematic Literature Review of Skyline Query Processing Over Data Stream. *IEEE Access*, *11*, 72813–72835. https://doi.org/10.1109/access.2023.3295117

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. Proceedings of the Twenty-First International Conference on Machine Learning, 78. https://doi.org/10.1145/1015330.1015435

Papadias, D., Tao, Y., Fu, G., & Seeger, B. (2005). Progressive skyline computation in database systems. *ACM Transactions on Database Systems*, *30*(1), 41–82. https://doi.org/10.1145/1061318.1061320

Prechelt, L. (2002). Early stopping — but when? In G. B. Orr & K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, 55–69. https://doi.org/10.1007/3-540-49430-8_3

Putten, P., & van Someren, M. (2000). CoIL 2000 insurance company benchmark [Data set]. UCI Machine Learning Repository. https://doi.org/10.24432/C5630S

Ren, W., Lian, X., & Ghazinour, K. (2019). Skyline queries over incomplete data streams. *The VLDB Journal*, *28*(6), 961–985. https://doi.org/10.1007/s00778-019-00577-6

Russell, S. J. (2010). Artificial Intelligence a Modern Approach. *London, U.K.: Pearson*.

Saad, N. H. M., Ibrahim, H., Sidi, F., Yaakob, R., & Alwan, A. A. (2021). Efficient Skyline Computation on *Uncertain Dimensions*. *IEEE Access*, *9*, 96975–96994. https://doi.org/10.1109/access.2021.3094547

Saleem, M. A., Javeed, A., Akarathanawat, W., Chutinet, A., Suwanwela, N. C., Asdornwised, W., Chaitusaney, S., Deelertpaiboon, S., Srisiri, W., Benjapolakul, W., & Kaewplung, P. (2024). Innovations in Stroke Identification:

A Machine Learning-Based Diagnostic Model Using Neuroimages. *IEEE Access*, 1–1. https://doi.org/10.1109/access.2024.3369673

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. IEEE Transactions on Neural Networks, 20(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. https://jmlr.org/papers/v15/srivastava14a.html

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An Introduction. The MIT Press.

Tan, K.-L., Eng, P.-K., & Ooi, B. C. (2001). *Efficient Progressive Skyline Computation*. 301–310.

Veličković, P. (2023). Everything is connected: Graph neural networks. Current Opinion in Structural Biology, 79(2301.08210.). https://doi.org/10.1016/j.sbi.2023.102538

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1710.10903

vom Brocke, J., Hevner, A., & Maedche, A. (2020). Introduction to Design Science Research. Progress in IS, 1–13. https://doi.org/10.1007/978-3-030-46781-4_1

Wang, H., Wang, W., Liu, Y., & Alidaee, B. (2022). Integrating Machine Learning Algorithms with Quantum Annealing Solvers for Online Fraud Detection. *IEEE Access*, 1–1. https://doi.org/10.1109/ACCESS.2022.3190897

Wang, Y., Shi, Z., Wang, J., Sun, L., & Song, B. (2017). Skyline Preference Query Based on Massive and Incomplete Dataset. *IEEE Access*, *5*, 3183–3192. https://doi.org/10.1109/access.2016.2639558

Yang, J., Yao, W., & Zhang, W. (2021). Keyword Search on Large Graphs: A Survey. *Data Science and Engineering*, *6*(2), 142–162. https://doi.org/10.1007/s41019-021-00154-4

Yiu, M. L., & Mamoulis, N. (2007). Efficient processing of top-k dominating queries on multi-dimensional data. *Proceedings 33rd International Conference Very Large Data Bases (VLDB)*, 483–494.

Yonis Gulzar, & Alwan, A. A. (2022). CIDS: An Efficient Algorithm for Processing Skyline Queries for Partially Complete Data in Cloud Environment. IEEE Access, 10, 66449–66466. https://doi.org/10.1109/access.2022.3185087

Zadeh, L. (1963). Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, *8*(1), 59–60. https://doi.org/10.1109/tac.1963.1105511

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. Advances in Neural Information Processing Systems (NeurIPS), 31, 5165–5175. https://doi.org/10.48550/arXiv.1802.09691

Zhang, S., Ray, S., Lu, R., Zheng, Y., Guan, Y., & Shao, J. (2022). Towards Efficient and Privacy-Preserving User-Defined Skyline Query over Single Cloud. IEEE Transactions on Dependable and Secure Computing, 1–1. https://doi.org/10.1109/tdsc.2022.3153790

Zhu, X., & Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review, 22(3), 177–210. https://doi.org/10.1023/B:AIRE.0000027336.62517.f8

APPENDIX I

GNN + ISKYLINE PSEUDOCODE

```
CLASS ISkyline:
    METHOD __init__(data, threshold = 10):
        SET self.data = data
        SET self.threshold = threshold

    METHOD dominates(p1, p2):
        SET all_gte = True
        SET any_gt = False
        FOR EACH (x, y) IN zip(p1, p2):
            IF x IS NOT None AND y IS NOT None:
                IF x < y:
                    SET all_gte = False
                IF x > y:
                    SET any_gt = True
        RETURN all_gte AND any_gt

FUNCTION compute_ground_truth_skyline(data):
    INITIALIZE skyline = []
    INITIALIZE labels = []
    FOR i FROM 0 TO LENGTH(data) - 1:
        SET point1 = data[i]
        SET dominated = False
        FOR j FROM 0 TO LENGTH(data) - 1:
            IF i != j:
                SET point2 = data[j]
                IF ISkyline([]).dominates(point2, point1):
                    SET dominated = True
```

```
            BREAK
    IF NOT dominated:
        APPEND point1 TO skyline
        APPEND 1 TO labels
    ELSE:
        APPEND 0 TO labels
  RETURN skyline, labels

# Main Execution

START memory tracking
START timer

LOAD CSV file INTO synthetic_data
SELECT columns ["Price", "Rating", "Availability", "Shipping_Time", "Category"]
FILL missing values with 0
CONVERT selected columns TO data (list of lists)

NORMALIZE data USING MinMaxScaler → data_normalized

CALL compute_ground_truth_skyline(data_normalized)
  → RETURN skyline_points, labels

CONVERT data_normalized, labels TO arrays

SEPARATE skyline_points WHERE label == 1
SEPARATE non_skyline_points WHERE label == 0

OVERSAMPLE skyline_points TO MATCH count of non-skyline_points
COMBINE oversampled skyline WITH non-skyline points INTO:
  balanced_data, balanced_labels

# Build Graph from Dominance Relations
```

```
INITIALIZE edges = []
FOR i IN RANGE LENGTH(balanced_data):
   FOR j IN RANGE LENGTH(balanced_data):
      IF i ≠ j AND ISkyline([]).dominates(balanced_data[i], balanced_data[j]):
         APPEND (i, j) TO edges

CONVERT edges TO tensor edge_index
CONVERT balanced_data TO tensor x (float)
CONVERT balanced_labels TO tensor y (long)

# Define GNN Model

CLASS GNN(input_dim, hidden_dim, output_dim):
   METHOD __init__():
      SET conv1 = GCNConv(input_dim, hidden_dim)
      SET conv2 = GCNConv(hidden_dim, hidden_dim)
      SET conv3 = GCNConv(hidden_dim, output_dim)
      SET dropout = Dropout(p = 0.5)

   METHOD forward(x, edge_index):
      x = conv1(x, edge_index)
      x = ReLU(x)
      x = dropout(x)
      x = conv2(x, edge_index)
      x = ReLU(x)
      x = dropout(x)
      x = conv3(x, edge_index)
      RETURN x

# Training Setup

SET input_dim = number of features
```

```
SET hidden_dim = 32
SET output_dim = 2

INITIALIZE model = GNN(input_dim, hidden_dim, output_dim)
INITIALIZE optimizer = Adam(model.parameters, lr = 0.005)
DEFINE loss_fn = BCEWithLogitsLoss(pos_weight = imbalance_ratio)

SPLIT data:
   train_mask = 80% random mask
   test_mask = remaining 20%

# Training Loop

FOR epoch FROM 0 TO 199:
   SET model TO training mode
   ZERO optimizer gradients
   COMPUTE out = model(x, edge_index)
   COMPUTE loss = loss_fn(out[train_mask], one_hot(y[train_mask]))
   BACKPROPAGATE loss
   optimizer.step()
   IF epoch MOD 10 == 0:
      PRINT "Epoch <epoch>, Loss: <loss>"

# Evaluation

SET model TO evaluation mode
DISABLE gradients:
   COMPUTE out = model(x, edge_index)
   COMPUTE probabilities = softmax(out[test_mask])[:, 1]
   SET predictions = 1 IF prob > 0.5 ELSE 0

GET y_true = y[test_mask]
```

COMPUTE precision, recall, f1, accuracy, auc_roc USING y_true and predictions

Performance Measurement

CALCULATE response_time = current_time - start_time
GET peak_memory FROM memory tracker
STOP memory tracking

Output Results

PRINT GNN Performance Metrics:
    Accuracy, Precision, Recall, F1 Score, AUC-ROC

PRINT Overall Performance:
    Query Response Time (seconds)
    Peak Memory Usage (KB)

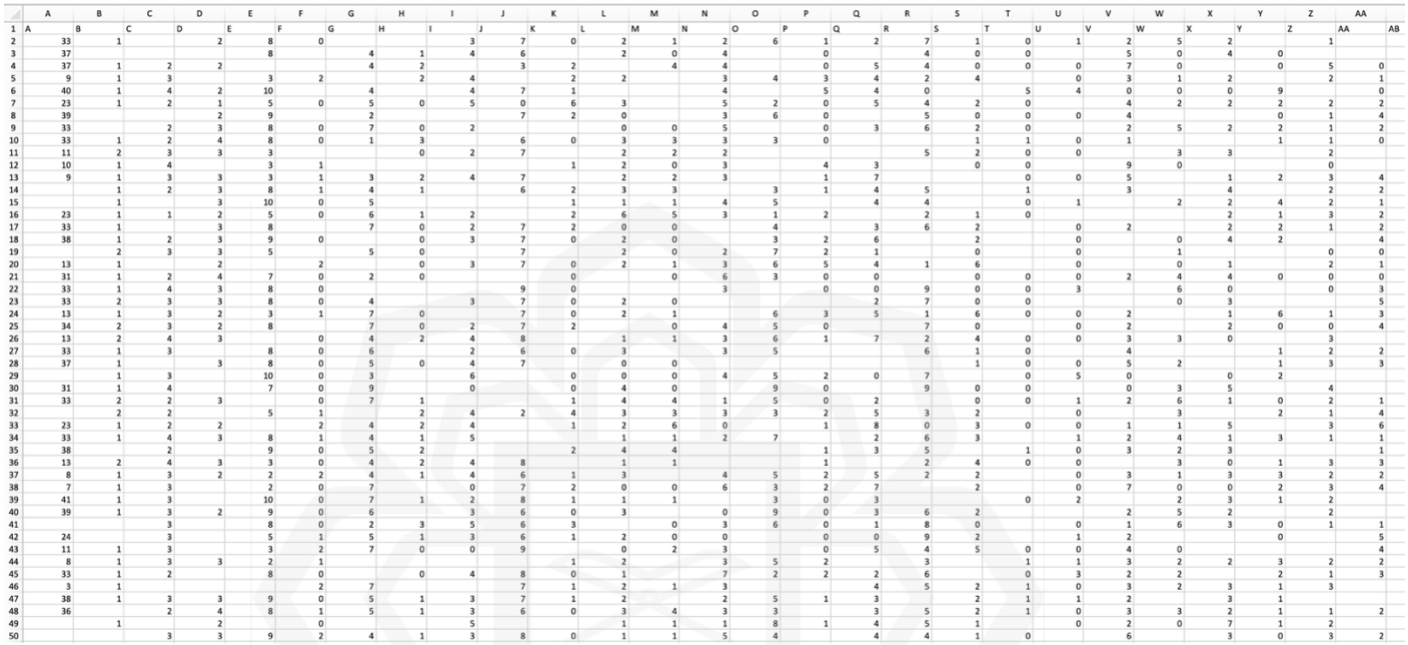
APPENDIX II

SAMPLE OF 10% INCOMPLETENESS SYNTHETIC DATASET

	A	B	C	D	E	F
1	Product_ID	Price	Rating	Availability	Shipping_Tin	Category
2	0	380.79	2.6	182	14	9
3	1	951.21	2.9	95	6	10
4	2	734.67	4.4	35	9	2
5	3	602.67	2.4	113	1	1
6	4	164.46		36	8	1
7	5	164.43	1.4	13	10	8
8	6	67.5	4.1	154	13	10
9	7	867.51	4.4		13	10
10	8	605.1		82	5	10
11	9	710.99	2.7	158	13	2
12	10	30.38	1.7	72	6	9
13	11		3.8	101	6	9
14	12	834.12	3.1		1	2
15	13	220.22	3.5	82	3	3
16	14			43	12	2
17	15	191.57	1.8	67	6	
18	16	311.2	1.2	47	14	7

APPENDIX III

SNIPPET OF COIL 2000 DATASET



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
33	1		2	8	0			3	7	0	2	1	2	6	1	2	7	1	0	1	2	5	2		1		
37				8		4	1	4	6		2	0	4		0		4	0	0		5	0	4	0			
37	1	2	2			4	2		3	2		4	4		0	5	4	0	0	0	7	0		0	5	0	
9	1	3		3	2		2	4		2	2		3	4	3	4	2	4		0	3	1	2		2	1	
40	1	4	2	10		4		4	7	1			4		5	4	0		5	4	0	0	0	9		0	
23	1	2	1	5	0	5	0	5	0	6	3		5	2	0	5	4	2	0		4	2	2	2	2	2	
39			2	9		2			7	2	0		3	6	0		5	0	0	0	4			0	1	4	
33		2	3	8	0	7	0	2			0	0	5		0	3	6	2	0		2	5	2	2	1	2	
33	1	2	4	8	0	1	3		6	0	3	3	3	3	0			1	1	0	1			1	1	0	
11	2	3	3	3			0	2	7		2	2	2				5	2	0	0		3	3		2		
10	1	4		3	1					1	2	0	3		4	3		0	0		9	0			0		
9	1	3	3	3	1	3	2	4	7		2	2	3		1	7			0	0	5		1	2	3	4	
	1	2	3	8	1	4	1		6	2	3	3		3	1	4	5		1		3		4		2	2	
	1		3	10	0	5				1	1	1	4	5		4	4		0	1		2	2	4	2	1	
23	1	1	2	5	0	6	1	2		2	6	5	3	1	2		2	1	0				2	1	3	2	
33	1		3	8		7	0	2	7	2	0	0		4		3	6	2		0	2		2	2	1	2	
38	1	2	3	9	0		0	3	7	0	2	0		3	2	6		2		0		0	4	2		4	
	2	3	3	5		5	0		7		2	0	2	7	2	1		0		0		1			0	0	
13	1		2		2		0	3	7	0	2	1	3	6	5	4	1	6		0		0	1		2	1	
31	1	2	4	7	0	2	0			0		0	6	3	0	0		0	0	0	2	4	4	0	0	0	
33	1	4	3	8	0				9	0			3		0	0	9	0	0	3		6	0		0	3	
33	2	3	3	8	0	4		3	7	0	2	0				2	7	0	0			0	3			5	
13	1	3	2	3	1	7	0		7	0	2	1		6	3	5	1	6	0	0	2		1	6	1	3	
34	2	3	2	8		7	0	2	7	2		0	4	5	0		7	0		0	2		2	0	0	4	
13	2	4	3		0	4	2	4	8		1	1	3	6	1	7	2	4	0	0	3	3	0		3		
33	1	3		8	0	6		2	6	0	3		3	5			6	1	0		4			1	2	2	
37	1		3	8	0	5	0	4	7		0	0						1	0	0	5	2		1	3	3	
	1	3		10	0	3		6		0	0	0	4	5	2	0	7		0	5	0		0	2			
31	1	4		7	0	9		0		0	4	0		9	0		9	0	0		0	3	5		4		
33	2	2	3		0	7	1			1	4	4	1	5	0	2		0	0	1	2	6	1	0	2	1	
	2	2		5	1		2	4	2	4	3	3	3	3	2	5	3	2		0		3		2	1	4	
23	1	2	2		2	4	2	4		1	2	6	0		1	8	0	3	0	0	1	1	5		3	6	
33	1	4	3	8	1	4	1	5			1	1	2	7		2	6	3		1	2	4	1	3	1	1	
38		2		9	0	5	2			2	4	4			1	3	5		1	0	3	2	3			1	
13	2	4	3	3	0	4	2	4	8		1	1					2	4	0	0		3	0	1	3	3	
8	1	3	2	2	2	4	1	4	6	1	3		4	5	2	5	2	2		0	3	1	3	3	2	2	
7	1	3		2	0	7		0	7	2	0	0	6	3	2	7		2		0	7	0	0	2	3	4	
41	1	3		10	0	7	1	2	8	1	1	1		3	0	3			0	2		2	3	1	2		
39	1	3	2	9	0	6		3	6	0	3		0	9	0	3	6	2			2	5	2		2		
		3		8	0	2	3	5	6	3		0	3	6	0	1	8	0		0	1	6	3	0	1	1	
24		3		5	1	5	1	3	6	1	2	0	0		0	0	9	2		1	2			0		5	
11	1	3		3	2	7	0	0	9		0	2	3		0	5	4	5	0	0	4	0				4	
8	1	3	3	2	1					1	2		3	5	2		3		1	1	3	2	2	3	2	2	
33	1	2		8	0		0	4	8	0	1		7	2	2	2	6		0	3	2	2		2	1	3	
3	1				2	7			7	1	2	1	3			4	5	2	1	0	3	2	3	1	3		
38	1	3	3	9	0	5	1	3	7	1	2		2	5	1	3		2	1	1	2		3	1			
36		2	4	8	1	5	1	3	6	0	3	4	3	3		3	5	2	1	0	3	3	2	1	1	2	
	1	2			0			5			1	1	1	8	1	4	5	1		0	2	0	7	1	2		
		3	3	9	2	4	1	3	8	0	1	1	5	4		4	4	1	0		6		3	0	3	2	

APPENDIX IV

SNIPPET OF NBA DATASET

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	G	GS	MP	FG	FGA	FG%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	PF	PTS	Trp-Dbl
2		80	3277	978	2173	0.45		819	0.85	71	354			147	233	2832	
3	79	79	3361	875	1823		601		0.738		481	556	521		181	2478	
4	72	72	3103	815	1822	0.447	675		0.814	44	188	232	532	140	121		0
5	80	80	3384	746	1668	0.447	655	799	0.82	59	221	280	484	161	286	2346	0
6	81	81	3089	751	1564	0.48	539	598	0.901	115	613		226	58	164	2151	0
7	80	80				0.481	573	709	0.808	122	272		216	88	229	2122	0
8	79	79	3084	689		0.471	627	812		77	453	530	375	107	223		
9			2892	699	1413	0.495	629	803	0.783		323	430		146	217	2040	2
10	80	80	3130		1516	0.45	501	571		78	264	342	229	95	157	2028	0
11	78	78		681	1500	0.454	324	359	0.903	71	261	332	286	105	151	1955	0
12	79	79		756	1435	0.527	440	568	0.775	236	554	790	208		227	1953	0
13	79	79	2906	653	1518	0.43	480	601		135	327	462	338	94	235	1911	1
14		81	3263	716	1365	0.525	241	298	0.809	249	710	959	143	160	223	1769	0
15	75	75	2877	641	1438	0.446	276	410	0.673	105	333	438	232	97	206	1741	
16			3167	597	1382	0.432	342	403	0.849	29	211	240	444	82	169	1728	
17	82		3288	660	1494		217	297	0.731	167	598	765	158	90	189	1684	
18		76	2957	626	1191	0.526	396	489	0.81	214		966	308			1656	
19		82	3340	632	1395	0.453	261	330		98	237	335	536	103	187	1653	1
20	80		3135	600	1194	0.503	425	617	0.689	191	522	713	371	46	184		1
21	80	80	2825	649	1322	0.491	256	303	0.845	84		256	275	52	207		0
22	79	79	2925	576	1228	0.469	283	338	0.837	45	217		460	72			0
23	70	70	2751	549	1087	0.505	474	581	0.816	204	443	647	181	50		1572	0
24	78	77		538	1151	0.467	350	428		111	279	390		102	191	1568	0
25		78	3059	495	1005		471	580	0.812			534	297	59	193	1521	0
26	75	75	2893	617	1422	0.434	263	348	0.756	184	557		256	103	219	1518	0
27		78		584	1304	0.448	275	345	0.797	65	286		394	91	190	1516	0
28	42	42	1747		709	0.464	129	164	0.787	40		187	221	49	91	827	
29			1460	255	595	0.429	146	181	0.807	25	139	164	173		99	689	0
30	80	80	2715	623	1136	0.548	253		0.707	38	223	261	460	83	161		0
31	81		2925	423	1012	0.418		520		41	211	252	699		160	1495	0
32	79		2796	541	1056	0.512	257		0.921	47	286	333				1489	1
33	80	80	2784	574	1185	0.484	335	533	0.629	231	650	881	253	70	219	1485	0
34		82	2945	519	1128		354	454	0.78	211	535	746	194	40	285	1472	0
35	76	76	2782	551	1220	0.452	243	350		132	391	523	238	85	301	1411	0
36		77	3140	478	1096	0.436	241	294	0.82	65	316		190	104	223	1374	0
37		80	2798	516	1099	0.47	168	210	0.8	31	127	158	306	100	196	1371	0
38		71	2729	493	1012	0.487	278	310	0.897	77	236	313	213	40	150	1366	0
39	40	40		292	590	0.495	163	182	0.896	36	154	190	112	21	90		0
40	32	31	1174	201	422	0.476		128	0.898	41	82	123		19	60	561	0
41	80	47	2482	486	1152	0.422	211	268	0.787	43	176	219	240	75		1349	
42	78	75		493	1112	0.443	289	335	0.863	40	247		491	74	225	1345	0
43	74	71	2545			0.436	245			193	399	592	144	57		1333	
44	81		2910	472		0.411	268	341	0.786	48	264	312	225	104	195	1329	
45	75	54	2708	494	1086	0.455	289	332	0.87	114	352	466	186	127	238		0
46	82	81	3021		881	0.531	356	598	0.595	288		1022	125	65	277	1292	0
47	71	71	2601	445	1018	0.437	238		0.915	79	336		136	45		1290	0
48	31	31	1147	171	424	0.403		105		34	131	165	68		82		0
49	40		1454	274		0.461	140	155	0.903	45	205	250	68	26	89	779	0
50	81	81	2955	451	1078	0.418	256	314	0.815	29	259	288	514	94	251		0

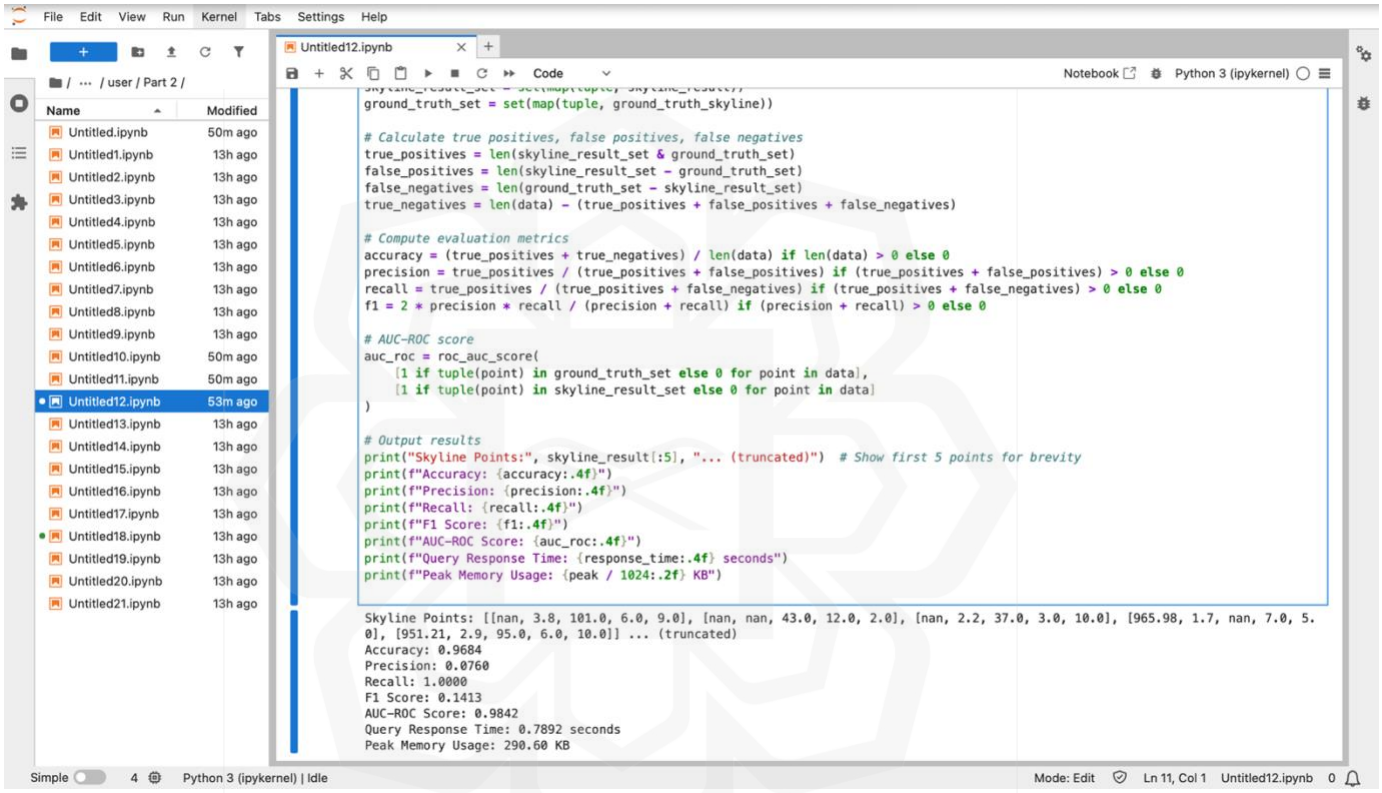
APPENDIX V

SNIPPET OF MOVIELENS DATASET

	A	B	C	D
1	userId	movieId	rating	timestamp
2			4	964982703
3		3	4	
4	1	6	4	964982224
5	1		5	964983815
6	1	50	5	964982931
7	1	70	3	
8	1	101	5	964980868
9	1	110		964982176
10	1		5	964984041
11	1	157	5	964984100
12		163	5	
13	1	216	5	964981208
14	1		3	964980985
15		231	5	964981179
16	1	235	4	964980908
17	1		5	964981680
18	1	296	3	964982967
19	1			964982310
20	1	333	5	
21	1	349	4	964982563
22	1	356	4	
23	1	362		964982588
24	1	367	4	964981710
25	1	423	3	964982363
26		441		964980868
27	1	457	5	
28	1	480	4	
29				964981208
30	1	527	5	964984002
31		543	4	964981179
32	1	552	4	964982653
33	1	553	5	964984153
34		590	4	964982546
35	1	592	4	964982271
36	1	593	4	964983793
37	1	596	5	964982838
38		608	5	
39	1	648	3	964982563
40	1	661	5	964982838
41	1	673	3	964981775
42	1	733	4	964982400
43	1	736	3	
44	1	780	3	964984086
45	1	804	4	964980499
46		919	5	964982475
47		923	5	964981529
48	1	940		964982176
49	1	943	4	964983614
50	1	954	5	964983219

APPENDIX VI

SKYLINE POINTS FOR PROCESSING OIS USING 7.5% INCOMPLETENESS SYNTHETIC DATASET



GLOSSARY

Adaptive Query Strategy. An approach to dynamically adjust query processing techniques to improve efficiency, particularly in handling large-scale or incomplete datasets.

Artificial intelligence. The conceptualization and creation of computational systems capable of performing tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

Bucket Algorithm. A technique for skyline queries that divides data into distinct subsets (buckets) based on shared attributes to simplify dominance computations.

Database. An organized collection of data stored digitally, designed to be easily retrieved, managed, and accessed through different methods.

Dominance Relationship. A relationship where one data point is considered superior to another if it performs as well or better across all dimensions and strictly better in at least one.

Dynamic Databases. Databases that undergo frequent updates, insertions, or deletions, requiring adaptable query algorithms.

Graph Neural Networks (GNNs). Machine learning models designed for graph-structured data, utilizing message-passing and aggregation techniques to capture relationships between nodes.

Incomplete Graphs. Graphs where some nodes or edges have missing attributes or data, requiring specialized handling techniques for accurate computations.

ISkyline Algorithm. An optimized skyline computation method that uses virtual points and shadow skylines to reduce the number of comparisons in datasets with missing values.

Machine Learning (ML). A subset of artificial intelligence focused on creating algorithms that learn from data to improve performance on specific tasks.

Pareto Optimality. A principle where a state is optimal if no improvement in one aspect can be achieved without compromising another aspect, commonly used in skyline queries.

Query. A question, especially one expressing doubt or requesting information.

Reinforcement Learning (RL). A machine learning paradigm where algorithms learn to make decisions by receiving rewards or penalties based on their actions.

Skyline Queries. Queries that filter datasets to return points that are not dominated by others, identifying optimal or preferred data points based on user-specified criteria.

Virtual Points. Artificially introduced points in algorithms like ISkyline to handle missing data and reduce dominance comparisons.

INDEX