Performance Analysis of K-Nearest Neighbor Algorithms

Heli Alpeshkumar Patel
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
helialpeshkumarpatel@cmail.carleton.ca

December 8, 2022

Abstract

The most popular machine learning algorithm is K nearest neighbour because it is widely applied across a wide range of fields and serves a variety of functions while delivering excellent results. As a model-free lazily learning approach, K nearest neighbour has some limitations because of its computational complexity. In contrast to existing model-based classification algorithms that build a model with a given training dataset and predict any test samples using the generated model, KNN classifier requires to store all the training instances in memory in order to locate all K nearest neighbours for a test sample.In the course of this study, I addressed this problem, compared the performance of sequential KNN algorithm and parallel KNN algorithm implementations, and at the end, came to certain conclusions. By the end, it was clear that using Parallel KNN algorithm when working with massive data will improve performance in terms of speed and efficiency. To conclude, the Sequential KNN algorithm is outperformed by the MPI multiprocessing implementation for parallelizing KNN algorithm.

1 Introduction

The amount of various datasets has significantly increased in recent years. There is a great need for innovative techniques that can transform these enormous quantities of data into useful information automatically. The intriguing, important, and comprehensible patterns that data mining reveals in huge datasets. Important application domains include business intelligence, customer relationship management, e-commerce, the World Wide Web, scientific simulation, and many more.

Higher intelligence is needed to deal with the present expansion of Big Data because standard computer performance cannot analyze this enormous volume of data. High-performance computing (HPC) must be taken into account while designing and developing systems since it can speed up calculation and produce precise results at a lower cost. There are several different types of high-performance computing, including computer clusters, grid computing, cloud computing, graphic processing units (GPU), MIC, and FPGA. One technique for achieving great performance on the GPU level is distributed memory programming with several processors. Parallelizing machine learning algorithms is one technique to use high performance computing to manage Big Data[14].

Classification is one of the core techniques in data mining, involving the training of a model on a dataset containing class labels and using the output to infer the class of unlabeled objects. The K-Nearest Neighbors (KNN) technique is one of the most popular machine learning algorithms since it makes categorization simple and effective[21][30]. KNN is a technique that classifies an item by using the training set's nearest point[21]. Since the KNN method is easy to use, quick to implement, and has a low error rate, it can effectively manage noisy data sets and produce accurate results. These are the rationales behind its application in such a wide range of fields[14][24].

High-dimensional data sets can be used with the K-Nearest Neighbor technique since its computation is straightforward. However, when the test set, train set, and data dimension are higher than anticipated, the computational complexity will be quite high and the operating time will be extremely long [20]. To address the problem parallel processing can be utilized and the time complexity can be significantly reduced with the use of parallel processing.

A standardised method of communicating across many computers executing concurrent programs via distributed memory is the message passing interface (MPI)[13]. When a message is sent to an object, parallel process, subroutine, function, or thread, it is usually referred to as "passing a message," and that message is then utilised to start a different process.

The goal of this study is to analyze the performance of Serialized KNN and Parallelized KNN. I have utilized Message Passing Interface(MPI) to parallelize KNN algorithm. I have made use of openly available dataset for both implementations to discover the performance differences in terms of speedup and efficiency. Based on the obtained performance results I would like to draw some conclusions and to see which one is better.

The study is organized as follows: initially, I've described the cutting-edge K-Nearest Neighbor algorithm, including how it works, what applications it may be used for, and what limitations it has. I later covered the parallel KNN technique. The issue raised throughout the study is detailed in sections 3 and 4, along with a possible resolution. Additional sections include Experimental Evaluation in Section 5 and Experiments and Results in Section 6. Section 7 discusses the Conclusion in detail.

2 Literature Review

2.1 Overview of K Nearest Neighbour

Based on supervised learning, one of the most basic machine learning techniques, is K-Nearest Neighbors. K-NN is a non-parametric method that makes no assumptions about the underlying data. It is also known as a lazy learner algorithm since it saves the training dataset rather than learning from it immediately. Instead, when classifying data, it performs an action using the dataset. The KNN algorithm merely stores the data during the training phase, and when it receives new data, it categorises it into a category that is quite similar to the new data [6].

The K Nearest Neighbor method, which is used for regression and classification, is categorized as supervised learning algorithm[12]. It is most frequently used as a classification technique since it relies on the idea that related points can be discovered close to one another [17] [11].

KNN has been used to tackle a variety of problems, including the following, in many different contexts. The KNN algorithm, which is a part of machine learning, was used to fill in the gaps in N Saranya's proposed solution for diagnosing chronic kidney disease in the year 2021 [26]. Sumandeep et al. (2018) [22] used a KNN classifier to work on sentiment analysis. Sentiment analysis is a method created to examine the positive, negative, and neutral elements of any text unit. In recent years, numerous algorithms have been created for the sentiment analysis of twitter data. In their research work, they describe a novel method for the sentiment analysis of tweet data that is based on a prior study about sentiment analysis. The suggested strategy combines techniques for feature extraction and categorization. The N-gram algorithm is used to extract features, and a KNN classifier is used to divide the input data into positive, negative, and neutral categories. Face recognition has long been a hot topic, particularly now that Covid-19 is so prevalent and less physical touch is required in settings where personnel identification is crucial. In 2021, researchers [19] assessed the effectiveness of K-Nearest Neighbors (KNN) for facial recognition in several scenarios. The experimental findings show that K-Nearest Neighbors (KNN) performed better than other methods. It is important to note that the accuracy of the KNN classifier for face recognition is 100% for frontal faces that are exposed and 74.7% for those that are covered.

2.1.1 Classification

A class label is chosen for classification issues based on a majority vote, meaning that the label that is most commonly expressed around a particular data point is adopted[11]. Let's think about the case displayed in Figure 1. There are two classes, A and B, and nearby data points will be used to categorize orange coloured point to identify whether that orange point belongs to A or B by calculating their distance from the k closest data points.

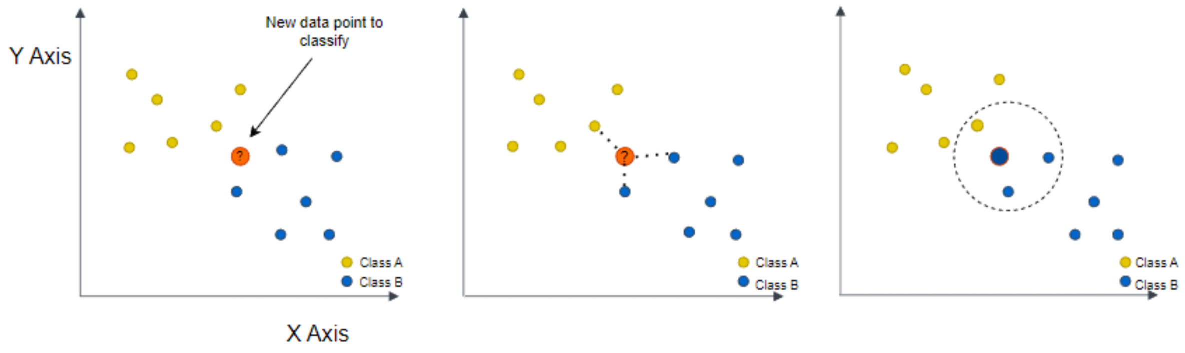


Figure 1: KNN diagram

2.1.2 Regression

Similar to classification issues, regression problems utilize the same principle; however, in this instance, a classification prediction is made by averaging the classifications of the first k nearest neighbors[11].

2.2 Euclidean Distance

The most common distance metric used by KNN for distance calculations is the Euclidean distance function. This method of measuring distance is mostly used to determine the separation between adjacent places. It is typically employed to measure the separation between two vectors with real values. When calculating the distance between real numbers such as integers, floats, and other types, we utilise the Euclidean distance [6][2].

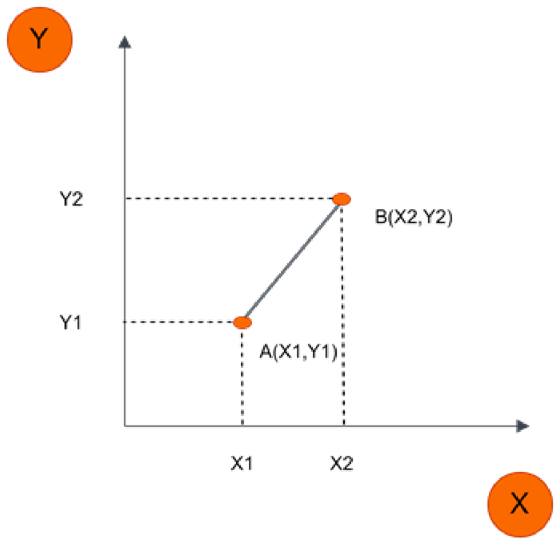


Figure 2: Euclidean distance between A and B

The Euclidean distance between A and B in 2 can be defined as follows:

$$= \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

2.3 Serialized implementation of KNN

Based on the following method, the operation of the K-NN can be explained[16][5]:

Step 1: It is to choose the neighbours' K-number.

Step 2: Determine the K-number of neighbours' Euclidean distances from each other.

Step 3: Using the Euclidean distance estimate, select the K closest neighbours.

step 4: Count the amount of data points in each category among these k neighbours.

Step 5: Place the new data points in the category where the number of neighbours is highest.

Step 6: Our model is complete.

2.4 Laziness of K-Nearest Neighbour and previous research proposing solutions

KNN classifier is a model-free lazily learning algorithm. In contrast to current model-based classification algorithms, which build a model with a given training dataset and predict any test samples using the generated model, KNN classifier requires to store all the training instances in memory in order to locate all K nearest neighbours for a test sample. [27][30].

A straightforward yet effective machine learning classification method is the K-Nearest Neighbors (KNN) algorithm. However, it has some flaws, including a large memory requirement, a slow processing speed, class overlap, and a challenging K value setting process.To address the aforementioned problems in a single framework, xin zhang et al.[30] proposed an Improved K-Nearest Neighbor rule integrating Prototype Selection and Local Feature Weighting (IKNN PSLFW). Moreover, there are other studies are done as well for the improvement in efficiency of KNN such as, authors proposed a novel kNN algorithm with data-driven k parameter computation called S-KNN which is better in comparision of state-of-the-art KNN [28]. Then,other study presented a kTree technique that incorporates a training stage into the kNN classification to learn several optimal k values for various test/new samples [29].

According to me, it is obvious that while each of these methods has enhanced KNN functionality, they have not sufficiently solved the problem. There are research that show parallel processing significantly improved KNN effectiveness in various domains such as, using the popular K-Nearest Neighbor technique on multi-core CPUs, P.P. Halkarnikar et al. [20] presented a case study for categorising a big database, such as the electoral data for the Kolhapur constituency, into age-based categories. The three age groups for election candidates were YOUNG, MIDDLE, and OLD. The processing time produced a significant enhancement. In a later study in 2019 [25], the parallel gwk-NN classifier that the researchers had developed throughout the model training and image classification stages was further developed using calls to MPI and GDAL in the C++ development environment on an eight-core CPU. Early experiments by the authors show that the proposed parallel gwk-NN classifier can improve the performance of high-resolution remotely sensed images with different types of land cover.In the sections that follow, I will go over the parallel KNN implementation, experiments, and results.

2.5 Message Passing Interface(MPI)

In parallel computing, nodes are groups of computers or even individual processing cores on a single computer. A fraction of the entire computing issue is generally worked on by each node in a parallel configuration. To synchronise each parallel node's activities, communicate data across nodes, and exert command and control over the entire parallel cluster is the next difficult task. For these responsibilities, the message passing interface specifies a standard set of functions [13]. Benefits of the message passing interface include standardization, portability, speed, and functionality.

2.5.1 MPI terminology: Key terms and instructions

A few fundamental MPI ideas and commands are listed below [15][13]:

- Communicatior: These MPI communicator objects link several process groups. A contained process receives an independent identifier from a communicator command, which arranges it as an ordered topology. For instance, MPICOMMWORLD is a command for a base communicator.

- Key: A key determines the rank or order of a procedure in the communicator. The order is determined by the process's position in the communicator if two processes are assigned the same key.

- Point-to-Point: This communicates between two particular processes. Two popular blocking methods for point-to-point messaging are MPI Send and MPI Recv. Blocking is the practise of having the sending and receiving systems hold off on sending and finishing a message until a whole message has been appropriately transmitted and received.
- Basic collective operations: These require communication between every process in a process group. One such method is MPIBcast, which distributes data from one node to all processes in a process group.

2.5.2 mpi4py

Python programs can use many processors since MPI for Python offers MPI bindings for the language. The object-oriented interface provided by this package closely resembles the MPI-2 C++ bindings and is based on the MPI specification [8].

2.6 Parallelized approach for K-Nearest Neighbour

Below is a description of the parallel K-Nearest Neighbour algorithm, which uses a master-slave design [18][10].

- Step 1: Load the training dataset, the test dataset, and obtain value K.
- Step 2: Choose 1 processor to be the master, while the remaining N-1 processors will serve as slaves.
- Step 3: These are the tasks that Mater carries out: The master separates the training samples into N subsets and distributes one subset to each processor together with the test dataset and value k, while reserving one subset for local processing (Master participates in distance computation too).
- step 4: The next step is performed by slave processors, who now compute the distance measurements independently, store them in a local array. This list's items are all tuples (distance, label). Each processor notifies the master when it has done processing distance computations by sending a message to it. Send the Master the local results for the complete test dataset.
- Step 5: Next, the master node performs the following:

  Concatenate all local results into a single list and sort it in ascending order after receiving the local results from the workers. There will be a separate only list for each test data point. To determine accuracy, perform the following steps for each test data point: Globally select the first k elements from the only list Compute the majority of labels/classes among the global k elements to obtain the predicted label. Compare the predicted labels and the true labels of the entire test dataset.

Below in figure 3 flowchart of Parallel KNN algorithm is explained:

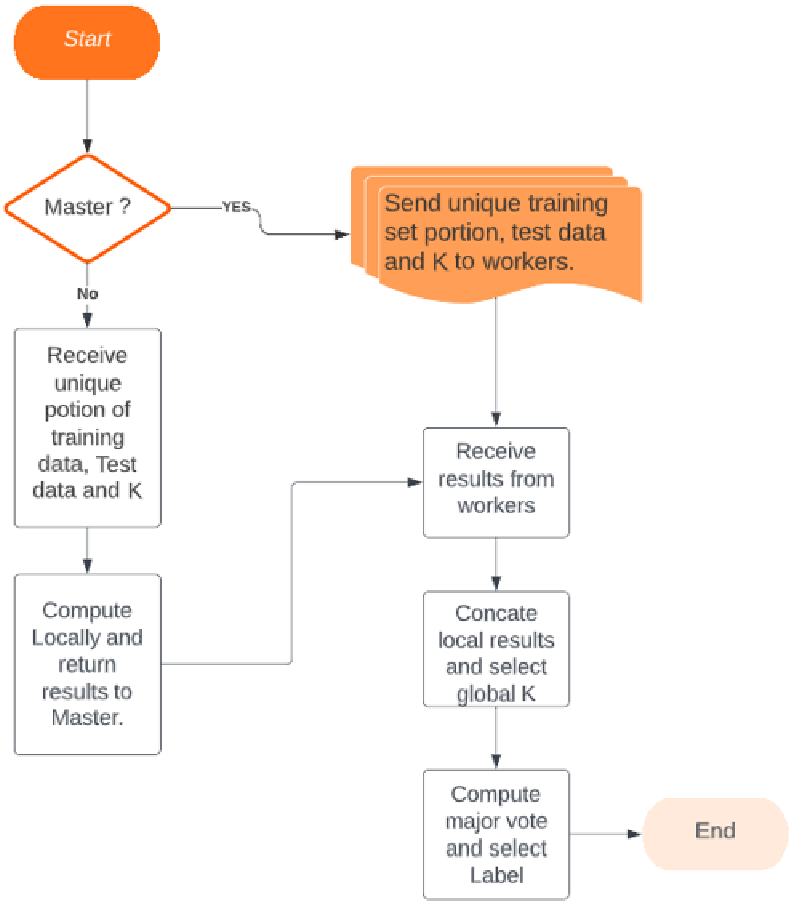

Start
Master ?
YES
No
Send unique training set portion, test data and K to workers.
Receive unique potion of training data, Test data and K
Compute Locally and return results to Master.
Receive results from workers
Concate local results and select global K
Compute major vote and select Label
End

Figure 3: Parallel KNN Algorithm

3 Problem Statement

As a result of the quick growth of hardware and software, databases can easily reach sizes of hundreds of Gigabytes. Typically, serial programs might take a very long time [23].

Due to its ease of use and effectiveness in classifying data, K-Nearest Neighbors (KNN) is one of the most popular machine learning algorithms [21][30]. KNN is a technique that classifies an item by using the training set's nearest point. The technique handles noisy datasets, produces useful results, and is regarded as straightforward. But the algorithm's drawbacks include increased time complexity and a challenging computation requirement [14].

As a result, the goal of this study is to analyze the performance of the state-of-the-art K-Nearest Neighbour and Parallel implementation of KNN algorithm. The KNN is parallelized using the MPI framework. The evaluations are performed in terms of the speed up and improved efficiency.

4 Proposed Solution

In this project, I tackled the KNN performance issue whereby computation time dramatically increases as data amount does. Since KNN is a crucial technique, I concentrated on the method's speed and efficiency during this project. Additionally, there is a noticeable increase in data generation nowadays, thus we must focus on KNN in order for it to function well and KNN can be utilized for Big Data problems.

It is interesting to note that while using state-of-the-art KNN I experienced challenges with time complexity despite seeing outstanding prediction accuracy. Thus, the problems with serialized KNN can definitely be improved with parallel processing.

I used MPI(Message Passing Interface) to parallelize the KNN algorithm. The purpose of this study is to analyze the speedup and efficiency of serialized and parallel KNN. I made use of the genuine dataset for credit card fraud detection[3]. I primarily focused on the performance variances in situations with varying computational complexity and dataset quantity for this performance analysis.

5 Experimental Evaluation

5.1 Experiment

With the two cases listed below, I used the master-slave data input decomposition algorithm for my project.

1. K = 5 was specifically chosen as the number of closest neighbours, and various training dataset sizes (ranging from 1000 to over 200000 records) and multiprocessor counts were examined(Processor counts=2,4 and 7).

2. On a training dataset of 10,000 records, examine with various values of k from 5 to 25 nearest neighbours, using various multiprocessor counts(Processor counts=2,4 and 7).

5.2 Environment

For MPI Implementation, a physical device with the following configuration is used:

- Device: Dell G3 3590
- Processor Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
- 16.0 GB of installed RAM (15.8 GB usable)
- OS: 64-bit windows 11

Programming language: Python
Following libraries were also utilized:

- Python - Pandas:
  An open-source toolkit called Pandas allows users to load large amounts of data from files like text or csv and then alter that data using a special data structure called a Dataframe[1].

- Python - Numpy:
  Array manipulation is done using the NumPy Python library. Additionally, it provides functions for working with matrices, the Fourier transform, and the linear algebra domain [9].

- Matplotlib:
  Python's Matplotlib toolkit provides a comprehensive tool for creating static, animated, and interactive visualizations [7].

5.3 Data

It is necessary to input real data with a significant number of records and features in order to conduct the experiment for KNN algorithms.

The information is kept in csv format files and is organised into training and test sets for the purpose of detecting credit theft. There are 56k test entries and 200k training entries [3].

5.4 Evaluation Metrics

To accurately analyze the parallel algorithm's performance in order to determine whether its use is practical or not. I utilised the performance indices listed below: Speedup and efficiency [4]

5.4.1 Speedup

The metric that illustrates the advantages of parallel problem solving is speedup. The ratio of the time needed to solve a problem on one processing element (TS) to the time needed to solve the same problem on p identical processing elements (Tp) is what is meant by this term.

$$S = \frac{T_s}{T_p}$$

5.4.2 Efficiency

Theoretically, a parallel system with p processing units may give us a speedup of p. But that doesn't happen very often. Usually, idle or communicating takes some time. Efficiency is a performance metric that assesses how efficiently processors are used in relation to the time and resources expended on synchronization and communication.

$$E = \frac{S}{p} = \frac{T_s}{pT_p}$$

6 Experiment and Results

6.1 Experiment 1: k=5 and data points from 1000 - 200k

With respect to Experiment 1, I kept Variable k (nearest neighbour to consider) = 5, and I then attempted parallel classification with a range of training dataset sizes and processors counts 2, 4 and 7. As the size of the data changed, we can see in the image below 4 how different implementations performed. The graph below shows that while a parallel implementation with 7 processor counts performs best, a sequential implementation takes the most time. It's intriguing that early on, when the size of the data points was less, there weren't much performance differences across the implementations.

I am confident in saying that when compared to sequential KNN, Parallel KNN's speed and efficiency have significantly improved. Parallel K-nearest neighbour operates more effectively and quickly the more processors there are. Additionally, I have listed in the table below 5 the parallel times that each processor experienced when the value of the data points varied, leading to changes in efficiency and speedup.

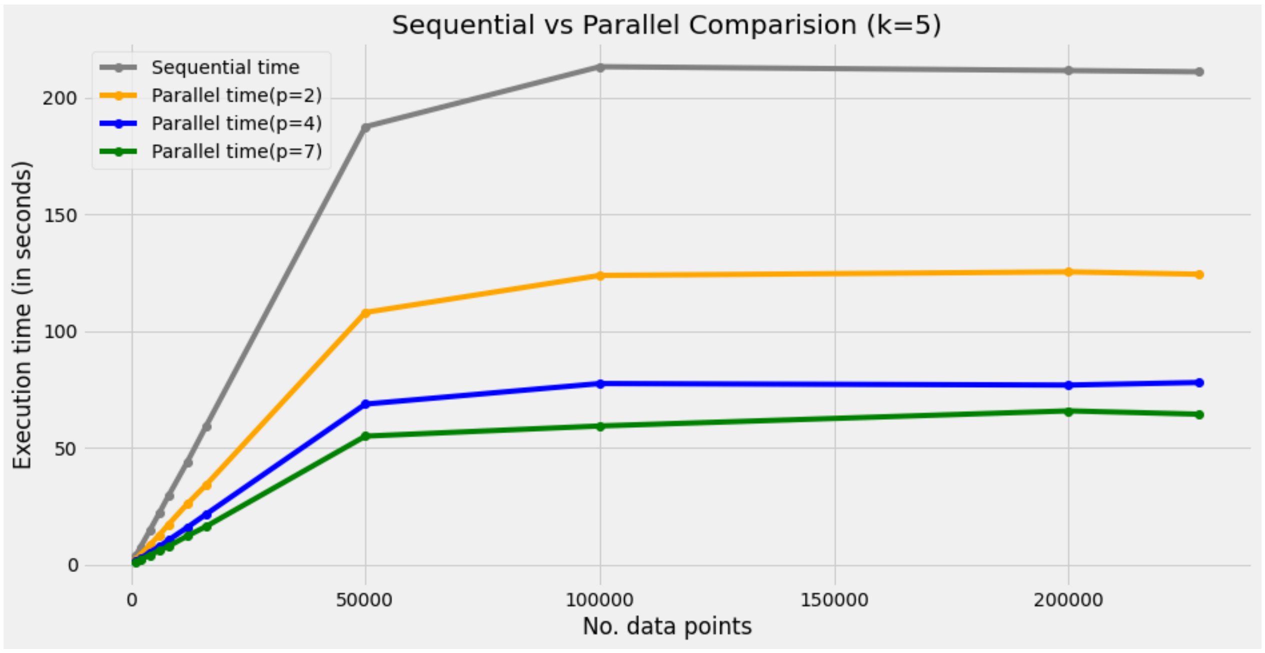
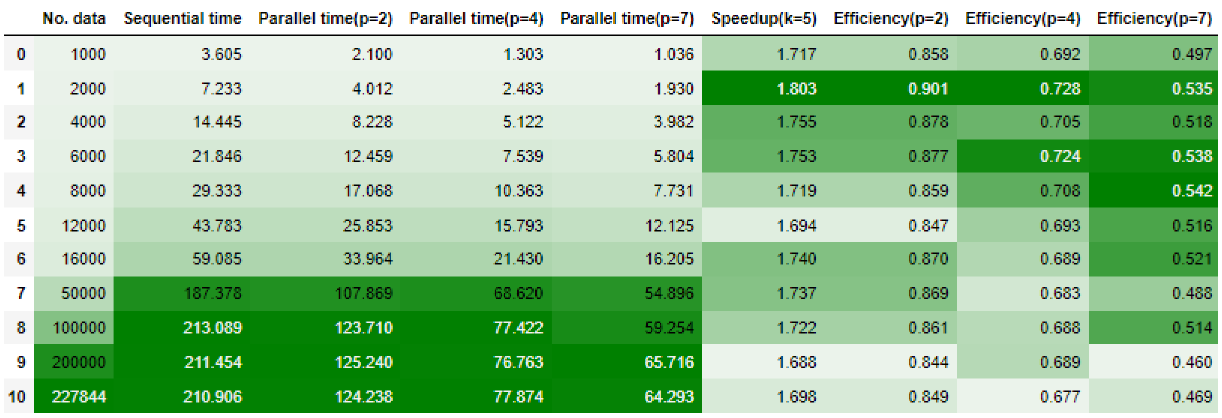


Figure 4: Range of data points and Multiprocessing



	No. data	Sequential time	Parallel time(p=2)	Parallel time(p=4)	Parallel time(p=7)	Speedup(k=5)	Efficiency(p=2)	Efficiency(p=4)	Efficiency(p=7)
0	1000	3.605	2.100	1.303	1.036	1.717	0.858	0.692	0.497
1	2000	7.233	4.012	2.483	1.930	1.803	0.901	0.728	0.535
2	4000	14.445	8.228	5.122	3.982	1.755	0.878	0.705	0.518
3	6000	21.846	12.459	7.539	5.804	1.753	0.877	0.724	0.538
4	8000	29.333	17.068	10.363	7.731	1.719	0.859	0.708	0.542
5	12000	43.783	25.853	15.793	12.125	1.694	0.847	0.693	0.516
6	16000	59.085	33.964	21.430	16.205	1.740	0.870	0.689	0.521
7	50000	187.378	107.869	68.620	54.896	1.737	0.869	0.683	0.488
8	100000	213.089	123.710	77.422	59.254	1.722	0.861	0.688	0.514
9	200000	211.454	125.240	76.763	65.716	1.688	0.844	0.689	0.460
10	227844	210.906	124.238	77.874	64.293	1.698	0.849	0.677	0.469

Figure 5: Overall Result for different range of data records

6.2 Experiment 2: 10k data points, k from 5 to 25

Next, Experiment has data points size as 10000, processors counts 2, 4 and 7 and value of K is ranging form 5 to 25. As a result for Experiment 2, the computation complexity rises in as K's value grows. We can see from the graph below 6 how effective parallel processing was in this situation.

Parallel processing definitely makes a difference in how quickly a task may be processed and completed when compared to sequential time, which is the highest time taking and worst performer. Additionally, as seen in the table below 7, more processors resulted in an improvement in the effectiveness and speed of our evaluation metrics.

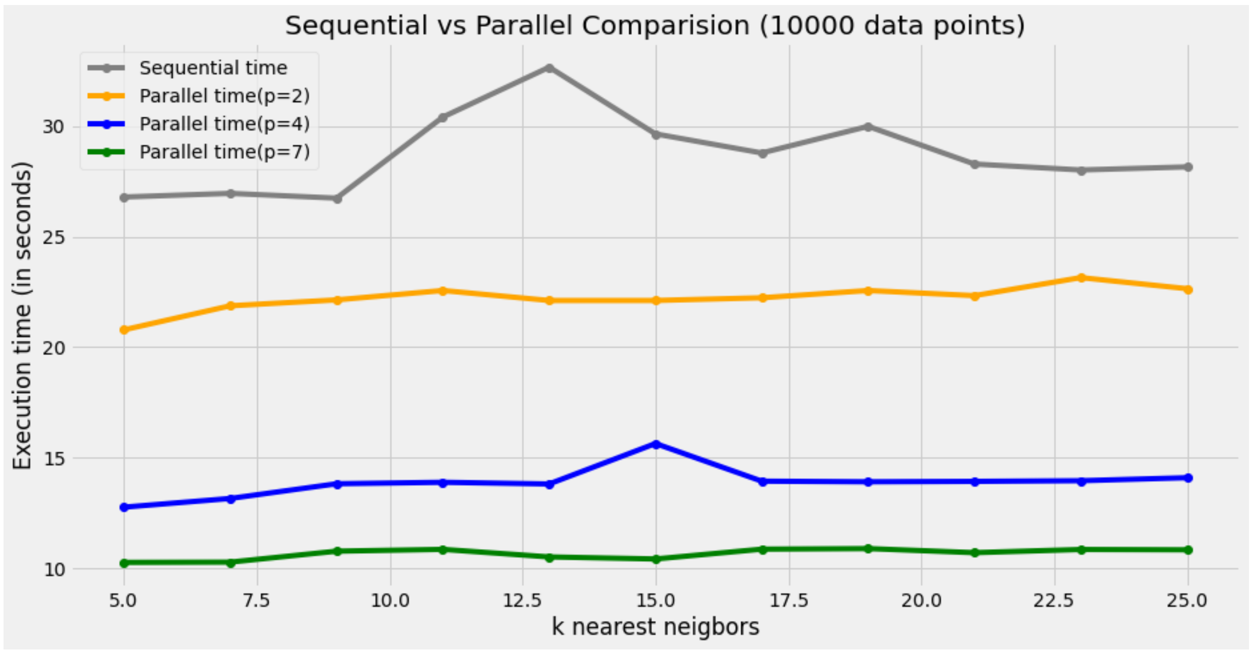
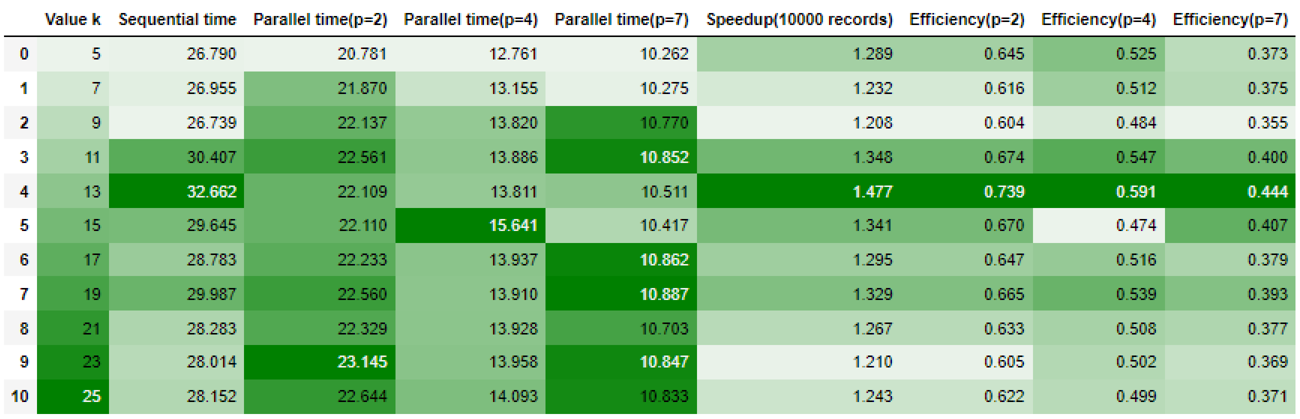


Figure 6: Range of k values and Multiprocessing



	Value k	Sequential time	Parallel time(p=2)	Parallel time(p=4)	Parallel time(p=7)	Speedup(10000 records)	Efficiency(p=2)	Efficiency(p=4)	Efficiency(p=7)
0	5	26.790	20.781	12.761	10.262	1.289	0.645	0.525	0.373
1	7	26.955	21.870	13.155	10.275	1.232	0.616	0.512	0.375
2	9	26.739	22.137	13.820	10.770	1.208	0.604	0.484	0.355
3	11	30.407	22.561	13.886	10.852	1.348	0.674	0.547	0.400
4	13	32.662	22.109	13.811	10.511	1.477	0.739	0.591	0.444
5	15	29.645	22.110	15.641	10.417	1.341	0.670	0.474	0.407
6	17	28.783	22.233	13.937	10.862	1.295	0.647	0.516	0.379
7	19	29.987	22.560	13.910	10.887	1.329	0.665	0.539	0.393
8	21	28.283	22.329	13.928	10.703	1.267	0.633	0.508	0.377
9	23	28.014	23.145	13.958	10.847	1.210	0.605	0.502	0.369
10	25	28.152	22.644	14.093	10.833	1.243	0.622	0.499	0.371

Figure 7: Overall Result for K ranging in 5 to 25

7 Conclusion

K-nearest neighbour is a crucial machine learning technique that has been used in numerous domains, therefore improving its speed and effectiveness will be helpful in a variety of contexts.

I have experimented with two scenarios, the first of which captures performance differences in terms of data record increment in size, and the second of which covers K values ranging from 5 to 25 for both the Sequential and Parallel KNN for the processors 2, 4, and 7. It was clear that when processor count increased, efficiency and speed got better. It is interesting to observe that while there is not much of a performance difference between Sequential and Parallel when the number of data points and value of k are lower, as the numbers increase, there is a significant performance difference, with Parallel KNN outperforming Sequential.

In this study, as I compared the performance of Serialized KNN algorithm and Parallel KNN algorithm, it became evident that Parallel KNN algorithm performed significantly better, faster and more effectively than Serialized KNN algorithm.

References

[1] 10 minutes to pandas. https://pandas.pydata.org/docs/user$_guide/10min.html.(Accessed: November 2022)$.

[2] 4 distance measures for machine learning. https://machinelearningmastery.com/distance-measures-for-machine-learning/. (Accessed: November 2022).

[3] Credit card fraud detection. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. (Accessed: November 2022).

[4] How to evaluate the performance of a parallel program. https://subscription.packtpub.com/book/application-development/9781785289583/1/ch01lvl1sec14/how-to-evaluate-the-performance-of-a-parallel-program/. (Accessed: November 2022).

[5] K-nearest neighbor. https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4. (Accessed: October 2022).

[6] K-nearest neighbor(knn) algorithm for machine learning. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. (Accessed: November 2022).

[7] Matplotlib: Visualization with python. https://matplotlib.org/. (Accessed: November 2022).

[8] mpi4py. https://mpi4py.readthedocs.io/en/stable/mpi4py.html. (Accessed: November 2022).

[9] Numpy: the absolute basics for beginners. https://numpy.org/doc/stable/user /absolute$_beginners.htmlnumpy-the-absolute-basics-for-beginners.(Accessed: November 2022)$.

[10] Parallel k-nearest neighbor. https://alitarhini.wordpress.com/2011/02/26/parallel-k-nearest-neighbor/. (Accessed: November 2022).

[11] Simple understanding and implementation of knn algorithm! https://www.ibm.com/topics/knn. (Accessed: October 2022).

[12] Simple understanding and implementation of knn algorithm! https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/, 2021. (Accessed: October 2022).

[13] message passing interface. https://www.techtarget.com/searchenterprisedesktop /definition/message-passing-interface-MPI, 2022. (Accessed: October 2022).

[14] Maha A. Alanezi and Abdulla AlQaddoumi. Applying parallel processing to improve the computation speed of k-nearest neighbor algorithm. In *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*, pages 1–6, 2020.

[15] Lyndon Clarke, Ian Glendinning, and Rolf Hempel. The mpi message passing interface standard. In *Programming environments for massively parallel distributed systems*, pages 213–218. Springer, 1994.

[16] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[17] Pádraig Cunningham and Sarah Jane Delany. K-nearest neighbour classifiers - a tutorial. *ACM Comput. Surv.*, 54(6), jul 2021.

[18] Reynaldo Gil-García, José Manuel Badía-Contelles, and Aurora Pons-Porrata. Parallel nearest neighbour algorithms for text categorization. In Anne-Marie Kermarrec, Luc Bougé, and Thierry Priol, editors, *Euro-Par 2007 Parallel Processing*, pages 328–337, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[19] Xinyu Guo. A knn classifier for face recognition. In *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 292–297, 2021.

[20] P. P. Halkarnikar, Ananda P. Chougale, H. P. Khandagale, and P. P. Kulkarni. Parallel k-nearest neighbor implementation on multicore processors. In *2012 International Conference on Radar, Communication and Computing (ICRCC)*, pages 221–223, 2012.

[21] S.B. Imandoust and Mohammad Bolandraftar. Application of k-nearest neighbor (knn) approach for predicting economic events theoretical background. *Int J Eng Res Appl*, 3:605–610, 01 2013.

[22] Sumandeep Kaur, Geeta Sikka, and Lalit Kumar Awasthi. Sentiment analysis approach based on n-gram and knn classifier. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 1–4, 2018.

[23] Shenshen Liang, Cheng Wang, Ying Liu, and Liheng Jian. Cuknn: A parallel implementation of k-nearest neighbor on cuda-enabled gpu. In *2009 IEEE Youth Conference on Information, Computing and Telecommunication*, pages 415–418, 2009.

[24] Chun Jie Ma and Zheng Sheng Ding. Improvement of k-nearest neighbor algorithm based on double filtering. In *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 1567–1570, 2020.

[25] Yingxia Pu, Xinyi Zhao, Guangqing Chi, Shuhe Zhao, Jiechen Wang, Zhibin Jin, and Junjun Yin. Design and implementation of a parallel geographically weighted k-nearest neighbor classifier. *Computers Geosciences*, 127:111–122, 2019.

[26] N Saranya, M Sakthi Samyuktha, Sharon Isaac, and B Subhanki. Diagnosing chronic kidney disease using knn algorithm. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 2038–2041, 2021.

[27] Shichao Zhang. Cost-sensitive knn classification. *Neurocomputing*, 391:234–242, 2020.

[28] Shichao Zhang, Debo Cheng, Zhenyun Deng, Ming Zong, and Xuelian Deng. A novel knn algorithm with data-driven k parameter computation. *Pattern Recognition Letters*, 109:44–54, 2018. Special Issue on Pattern Discovery from Multi-Source Data (PDMSD).

[29] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. Efficient knn classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1774–1785, 2018.

[30] Xin Zhang, Hongshan Xiao, Ruize Gao, Hongwu Zhang, and Yu Wang. K-nearest neighbors rule combining prototype selection and local feature weighting for classification. *Knowledge-Based Systems*, 243:108451, 2022.