**Project Title:**

**K-Nearest Neighbor Parallel (KNN-P) Algorithm**

A diagram of a diagram of a training

Description automatically generated with medium confidence

**Group Members:**

* Muhammad Hasnain (22K-5099)
* Ali Masood (22K-5127)
* Syed Shees Ali (22K-5047)

Instructor: Syed Faisal Ali

**Code Repository and Selection:**

* **Repository URL:**  [syedsheesali15/K-Nearest-Neighbour-Parallel-KNN-P-Algorithm-](https://github.com/syedsheesali15/K-Nearest-Neighbour-Parallel-KNN-P-Algorithm-)
* **Code Description:**

This project implements a parallelized version of the K-Nearest Neighbor (KNN) algorithm, aimed at reducing computation time through the distribution of distance calculations across multiple processors.

* **Complexity Analysis:**

The complexity increases as the number of test and training instances grows. Parallelization enables significant time savings, especially on larger datasets.

**Parallelization Strategy:**

* **MPI Implementation:**

- The KNN algorithm is parallelized using MPI. Each process calculates distances for a subset of the training dataset.

- Euclidean distances are computed independently across processes and results are gathered and sorted for final classification.

* **Code Segmentation:**

Key steps parallelized include:

1. Distribution of training data among processes.

2. Independent calculation of distances per process.

3. Gathering and sorting results for classification.

**Execution Plan:**

* **Hardware Specifications:**

- Processor: 12th Gen Intel(R) Core (TM) i7-12650H

- Clock Speed: 1.13 GHz

- RAM: 16 GB

* **Baseline Execution:**

Serial execution for 500 data points takes **3.8 ms**.

* **Parallel Execution:**

- Parallel execution with 3 processes reduces this to **2.8 ms**.

- Performance degradation observed beyond 50 processes due to excessive resource consumption.

**Data and Performance Metrics:**

* **Data Size:**

Training and test datasets with variable sizes will be utilized (e.g., 100–1000 instances).

* **Performance Metrics:**

- Execution time for serial and parallel implementations.

- Speedup and efficiency analysis for varying numbers of processes.

**Numerical Results and Visualization:**

* **Expected Results:**

Parallel implementation reduces computation time significantly for large datasets. Speedup is expected to plateau with an increasing number of processes.

* **Graphical Representation:**



Graphs showing execution times for serial and parallel implementations:

**- Red:** Serial Execution

**- Yellow:** Parallel (2 Processes)

**- Light Blue**: Parallel (4 Processes)

**- Dark Blue:** Parallel (8 Processes)

**Testing and Validation:**

* **Testing Methodology:**

- Validate correctness by comparing results of serial and parallel executions.

- Stress test on large datasets to ensure reliability.

* **Validation of Results:**

Execution times and classification accuracy will be compared to confirm the performance benefits of parallelization.

**Contribution of Group Members:**

* **Member Roles and Contributions:**

**- Ali Masood**: Algorithm implementation and MPI coding.

**- Muhammad Hasnain:** Performance testing and result compilation.

**- Syed Shees Ali:** Report writing and visualization and Algo implementation.

* **Collaboration Plan:**

Group members will jointly analyze results and refine the parallel implementation.

**Plagiarism and Originality Declaration:**

* **Originality Assurance:**

This project is original and leverages the KNN algorithm as a base to implement parallelization using MPI.

* **Use of AI Tools:**

AI tools were used to assist in formatting and testing the dataset.