**Student:** Hong Quan Doan

**Student ID:** 986956

**Day 12, 13 – Mar 09, 11**

**Environment**

MacBook, Core i5, 4 CPU, 8GB

**k-Nearest neighbor**

**Test results:**

Train: 39129

Test: 2059

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Serial Classifier - K: 10

Success: 1873

Mistakes: 186

Execution Time: 100.939 seconds.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Train: 39129

Test: 2059

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Number of threads: 1

Parallel Classifier Group - K: 10 - Factor 1 - Parallel Sort: false

Success: 1873

Mistakes: 186

Execution Time: 99.095 seconds.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Speedup: 1.019

Train: 39129

Test: 2059

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Number of threads: 2

Parallel Classifier Group - K: 10 - Factor 1 - Parallel Sort: false

Success: 1873

Mistakes: 186

Execution Time: 60.366 seconds.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Speedup: 1.672

Train: 39129

Test: 2059

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Number of threads: 3

Parallel Classifier Group - K: 10 - Factor 1 - Parallel Sort: false

Success: 1873

Mistakes: 186

Execution Time: 57.718 seconds.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Speedup: 1.749

Train: 39129

Test: 2059

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Number of threads: 4

Parallel Classifier Group - K: 10 - Factor 1 - Parallel Sort: false

Success: 1873

Mistakes: 186

Execution Time: 54.871 seconds.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Speedup: 1.840

**Source code:** (for Parallel version of lamda)

**public** String classify(Sample example) **throws** Exception {

**final** Distance[] distances = **new** Distance[dataSet.size()];

**final** CountDownLatch endControler = **new** CountDownLatch(numThreads);

**int** length = dataSet.size() / numThreads;

**int** remainder = dataSet.size() % numThreads;

**for** (**int** i = 0; i < numThreads; i++) {

**final** **int** startIndex = i \* length;

**final** **int** endIndex = (i == numThreads - 1) ? (i \* length + length + remainder) : (i \* length + length);

Runnable ltask = () -> {

**for** (**int** index = startIndex; index < endIndex; index++) {

Sample localExample = dataSet.get(index);

distances[index] = **new** Distance();

distances[index].setIndex(index);

distances[index].setDistance(EuclideanDistanceCalculator.*calculate*(localExample, example));

}

endControler.countDown();

};

executor.execute(ltask);

}

endControler.await();

**if** (parallelSort) {

Arrays.*parallelSort*(distances);

} **else** {

Arrays.*sort*(distances);

}

Map<String, Integer> results = **new** HashMap<>();

**for** (**int** i = 0; i < k; i++) {

Sample localExample = dataSet.get(distances[i].getIndex());

String tag = localExample.getTag();

results.merge(tag, 1, (a, b) -> a + b);

}

**return** Collections.*max*(results.entrySet(), Map.Entry.*comparingByValue*()).getKey();

}