1. **Algorithm:**

* **K – Nearest Neighbor**

**If K=1**

Step 1: Calculate the Euclidean distance between all the training test points with the test points and store it in the distance matrix.

Step 2: Find the training point which is nearest to the test point, which means the training point nearest to the test point.

Step 3: Assign the label of this training point to the test point.

**If K>1**

Step 1: Calculate the Euclidean distance between all the training test points with the test points and store it in the distance matrix.

Step 2: find the k training points which are nearest to the test point, which means the training points nearest to the test point.

Step 3: Assign the most numerous label of the k points to the test point.

* **Condensed 1- Nearest Neighbor**

Step 1: Initialize the subset with the single training data.

Step 2: Randomly choose another training data and perform the 1-nn algorithm. (In 1-nn pass the subset as the training and randomly chosen data as the test).

Step 3: Compare the result of this test with the actual training label.

Step 4: if both are same then discard the randomly chosen example else add the chosen example into the subset.

Step 5: Continue choosing the random example from the dataset until training data is empty or size of subset is equal to size of training set.

**Design decision for implementing C-NN**

I created the dummy index vector with the value 1 to the size of the training data to keep track of randomly chosen training data. I’m changing the value of this dummy index to zero after it chosen by random function. So, if sum of the value of all the index is zero, which means all the training examples are considered for the condensing.

1. **Observation:**

**K – Nearest Neighbor**

* **Table of accuracy with respect to value of K and N**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **K\Accuracy** | **N = 100** | **N = 1000** | **N = 2000** | **N = 5000** | **N= 10000** | **N = 15000** |
| **K=1** | 34.56 | 75.68 | 83.12 | 90.96 | 94.04 | 94.55 |
| **K=3** | 24.24 | 62.70 | 73.84 | 85.06 | 90.60 | 92.50 |
| **K=5** | 21.00 | 56.14 | 68.02 | 80.50 | 87.70 | 91.02 |
| **K=7** | 19.90 | 53.18 | 64.04 | 78.08 | 85.64 | 88.52 |
| **K=9** | 20.00 | 50.16 | 61.22 | 74.64 | 83.50 | 87.80 |

* **Table of running time with respect to value of K and N**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **K\Time** | **N = 100** | **N = 1000** | **N = 2000** | **N = 5000** | **N= 10000** | **N = 15000** |
| **K=1** | 0.014 | 0.10 | 0.21 | 0.49 | .094 | 1.24 |
| **K=3** | 0.030 | 0.21 | 0.37 | 1.00 | 2.25 | 3.35 |
| **K=5** | 0.023 | 0.18 | 0.39 | 0.99 | 2.10 | 3.56 |
| **K=7** | 0.026 | 0.21 | 0.41 | 1.06 | 2.17 | 3.48 |
| **K=9** | 0.028 | 0.20 | 0.38 | 1.03 | 2.22 | 3.18 |

**Condensed Nearest Neighbor**

* **Table of running time and # of rows in the subset with respect to value of N**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **N=100** | **N=1000** | **N=2000** | **N=5000** | **N=10000** | **N=15000** |
| **Time** | 0.06 | 0.49 | 0.72 | 2.18 | 5.00 | 8.47 |
| **# of rows in Subset** | 84 | 438 | 647 | 1160 | 1741 | 2245 |

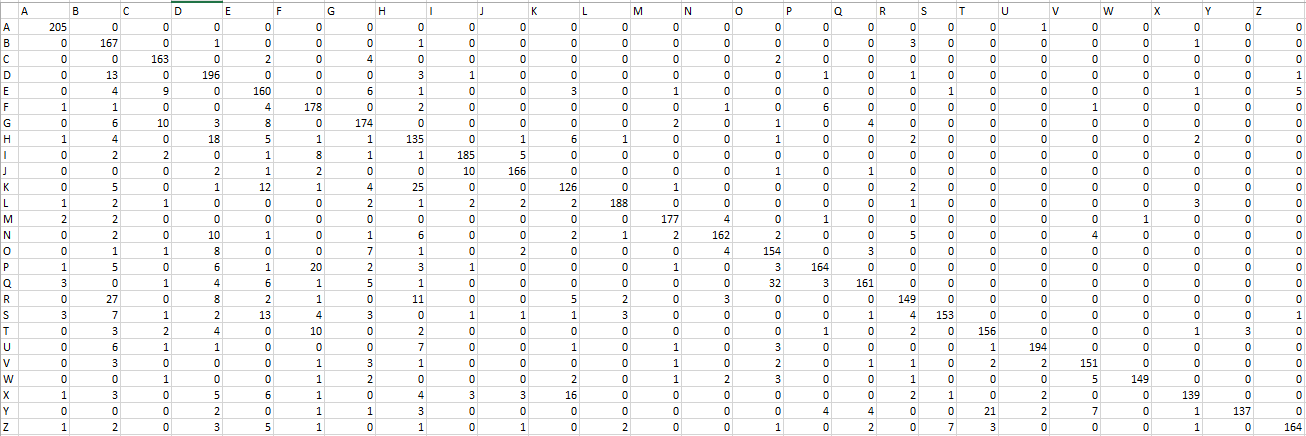
* **Table of accuracy with respect to value of K and N(running on the condensed subset)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K\Accuracy | N = 100 | N = 1000 | N = 2000 | N = 5000 | N= 10000 | N = 15000 |
| K=1 | 33.44 | 68.72 | 78.42 | 85.34 | 89.50 | 91.84 |
| K=3 | 22.66 | 44.92 | 52.56 | 60.22 | 66.86 | 68.56 |
| K=5 | 20.78 | 43.00 | 46.50 | 55.04 | 59.48 | 63.36 |
| K=7 | 18.92 | 40.52 | 45.82 | 52.00 | 56.62 | 59.68 |
| K=9 | 18.96 | 40.22 | 44.92 | 50.68 | 55.24 | 57.40 |

* **Table of running time with respect to value of K and N(running on the condensed subset)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K\Time | N = 100 | N = 1000 | N = 2000 | N = 5000 | N= 10000 | N = 15000 |
| K=1 | 0.02 | 0.04 | 0.09 | 0.11 | 0.21 | 0.23 |
| K=3 | 0.02 | 0.11 | 0.13 | 0.22 | 0.39 | 0.44 |
| K=5 | 0.03 | 0.11 | 0.15 | 0.23 | 0.32 | 0.48 |
| K=7 | 0.02 | 0.09 | 0.13 | 0.22 | 0.34 | 0.56 |
| K=9 | 0.02 | 0.09 | 0.12 | 0.24 | 0.37 | 0.46 |

**Confusion Matrix for the value of K = 3 and N = 5000**

****

1. **Discussion of various experiments**

* Accuracy of the algorithm decreases with the respect the value of N.
* Running time of the algorithm also decreases with the reduction of N.
* Accuracy of the algorithm also decrease as we increases the value of K.
* Running time remains constant while value of k is changing.